

Automatizace vytváření SIEM prostředí pro praktickou výuku bezpečnostních cvičení

Tomáš Trenz



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence
Akademický rok: 2024/2025

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	Tomáš Trenz
Osobní číslo:	A22059
Studijní program:	B0613A140020 Softwarové inženýrství
Forma studia:	Kombinovaná
Téma práce:	Automatizace vytváření SIEM prostředí pro praktickou výuku bezpečnostních cvičení
Téma práce anglicky:	Automating the Creation of SIEM Environments for the Practical Training of Security Exercises

Zásady pro vypracování

1. Zpracujte možnosti vytvoření SIEM prostředí dle zadání a proveďte specifikaci požadovaných funkcí.
2. Vyberte vhodné prostředky pro realizaci SIEM projektu.
3. Vypracujte návrh prostředí.
4. Proveďte implementaci v testovací infrastruktuře
5. Otestujte Vámi navržené řešení v kontextu zadání projektu.

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. HOCHSTEIN, Lorin, 2014. Ansible: up & running. Beijing: O'Reilly. ISBN 978-1-4919-1532-5.
2. KOLOUCH, Jan a BAŠTA, Pavel, 2019. CyberSecurity. CZ.NIC. Praha: CZ.NIC, z.s.p.o. ISBN 978-80-88168-31-7.
3. MILLER, David, c2011. Security information and event management (SIEM) implementation. New York: McGraw-Hill. ISBN 978-0-07-170109-9.
4. WOJŚŁAW, Damian a ADAMOWICZ, Grzegorz. The Linux DevOps handbook: customize and scale your Linux distributions to accelerate your DevOps workflow. Birmingham: Packt, [2023]. ISBN 978-1-80324-566-9.
5. BRIKMAN, Yevgeniy, 2022. Terraform: up & running: writing infrastructure as code. Third edition. Beijing: O'Reilly. ISBN 978-1-098-11674-3.

Vedoucí bakalářské práce: **Ing. David Malaník, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání bakalářské práce: **27. října 2024**
Termín odevzdání bakalářské práce: **2. června 2025**



doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan

L.S.

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 17. listopadu 2024

PROHLÁŠENÍ AUTORA BAKALÁŘSKÉ PRÁCE

Beru na vědomí, že

- odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb., v platném znění bez ohledu na výsledek obhajoby;
- bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému a bude dostupná k nahlédnutí;
- jedno vyhotovení bakalářské práce v listinné podobě bude ponecháno Univerzitě Tomáše Bati ve Zlíně k uložení;
- na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- podle § 60 odst. 1 autorského zákona má Univerzita Tomáše Bati ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- podle § 60 odst. 2 a 3 mohu užít své dílo – bakalářskou práci – nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tj. k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá; neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji, že

- jsem na bakalářské práci pracoval(a) samostatně a použitou literaturu jsem řádně citoval(a); v případě publikace výsledků budu uveden(a) jako spoluautor;
- odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou obsahově totožné.

Ve Zlíně, dne

.....
podpis autora

ABSTRAKT

Tato bakalářská práce se zabývá návrhem a implementací automatizovaného řešení pro nasazení SIEM prostředí. Cílem bylo vytvořit flexibilní, opakovatelné a výukově orientované prostředí pro praktické cvičení v oblasti kybernetické bezpečnosti. Řešení je postaveno na technologiích Packer, Terraform a Ansible a umožňuje automatizovaně vytvářet obrazy systému, nasazovat je do virtualizovaného prostředí VMware ESXi a konfigurovat komponenty SIEM včetně agentů, indexů, vstupních portů i uživatelských rolí. Součástí řešení je i podpora přenosné YAML konfigurace a testovací infrastruktura využitelná při výuce. Výsledný systém umožňuje studentům simulovat reálné scénáře sběru a analýzy dat a poskytuje prostor pro další rozšiřování.

Klíčová slova: SIEM, automatizace, Ansible, Terraform, Packer, výukové prostředí, kybernetická bezpečnost

ABSTRACT

This bachelor's thesis focuses on the design and implementation of an automated solution for deploying a SIEM environment. The goal was to create a flexible, repeatable, and educationally oriented platform for practical cybersecurity training. The solution is based on technologies such as Packer, Terraform, and Ansible, enabling the automated creation of system images, their deployment into a VMware ESXi virtualized infrastructure, and configuration of SIEM components, including agents, data inputs, indexes, and user roles. The system also includes support for portable YAML configuration and a testing environment suitable for classroom use. The outcome allows students to simulate real-world data collection and analysis scenarios while offering room for further extension.

Keywords: SIEM, automation, Ansible, Terraform, Packer, educational environment, cybersecurity

Věnování

Tuto práci věnuji svým rodičům Aleně a Josefovi, kterým se tímto omlouvám, že jsem během studií na ně neměl tolik času, kolik by si zasloužili. Zároveň jim děkuji za trpělivost, podporu a víru, že to zvládnu.

Poděkování

Děkuji také svým kolegům a projektovým vedoucím v práci za pochopení a umožnění flexibilní pracovní doby, díky které jsem mohl studium skloubit s praxí.

Velké poděkování patří vedoucímu práce, který mi umožnil věnovat se tématu blízkému mé profesi a dal mi prostor vytvořit prakticky zaměřené řešení s reálným přínosem.

Vážím si podpory všech přátel, kteří mě po celou dobu motivovali a fandili mi.

Zvláštní poděkování patří Vojtěšce a Vladimírovi – za laskavé zázemí během studijních víkendů, a především Lence, které děkuji za inspiraci a motivaci vrátit se po letech zpět na vysokou školu. Byly to tři náročné, ale krásné roky, které jsme sdíleli.

Poznámka autora

Prohlašuji, že při tvorbě této práce jsem použil nástroj generativního modelu ChatGPT od OpenAI (verze 4o), a to zejména při formátování dokumentu, strukturování kapitol, třídění myšlenek a jazykových úpravách. Po použití tohoto nástroje jsem provedl/a kontrolu obsahu a přebírám za něj plnou zodpovědnost.

OBSAH

ÚVOD.....	13
I TEORETICKÁ ČÁST.....	15
1 SECURITY INFORMATION AND EVENT MANAGEMENT.....	16
1.1 VÝZNAM SIEM.....	16
1.2 KLÍČOVÉ FUNKCE SIEM.....	16
2 PŘEHLED A SROVNÁNÍ SIEM ŘEŠENÍ.....	17
2.1 VÝBĚR PRODUKTŮ PRO SROVNÁNÍ.....	17
2.2 KRITÉRIA HODNOCENÍ.....	18
2.3 ANALÝZA JEDNOTLIVÝCH SIEM ŘEŠENÍ.....	20
2.3.1 Splunk Enterprise Security.....	20
2.3.2 Microsoft Sentinel.....	21
2.3.3 IBM QRadar.....	21
2.3.4 Securonix.....	21
2.3.5 Exabeam.....	21
2.3.6 Gurukul.....	21
2.3.7 Google Security Operations (dříve Google Chronicle).....	22
2.3.8 Devo.....	22
2.3.9 Elastic Security.....	22
2.3.10 OpenText ArcSight.....	23
2.3.11 LogRhythm.....	24
2.3.12 Rapid7 InsightIDR.....	24
2.4 VYBRANÝ SIEM.....	25
2.5 PROČ AUTOMATIZACE ANEB ÚVOD DO DEVOPS.....	27
2.5.1 Co je to DevOps.....	28
2.6 INFRASTRUKTURA JAKO KÓD (IAC).....	29
2.6.1 Skripty psané jednorázově.....	30
2.6.2 Nástroje pro správu konfigurace.....	30
2.6.3 Nástroje pro tvorbu šablon serverů.....	31
2.6.4 Nástroje pro orchestraci.....	31
2.6.5 Nástroje pro zřízení serveru.....	32
2.7 AUTOMATIZACE NASAZENÍ SIEM.....	32
2.7.1 Cíle automatizace.....	33
2.7.2 Uvažované technologie pro automatizaci.....	33
2.7.3 YAML konfigurace a jejich využití v Ansible.....	34
2.7.4 HashiCorp Configuration Language (HCL) v Terraformu.....	35
2.7.5 HCL v nástroji Packer.....	36
2.7.6 Konfigurační soubory .conf ve Splunku a podobnost s .ini.....	37
2.7.7 Shrnutí přehledu konfiguračních formátů a nástrojů.....	37
2.8 POŽADAVKY NA PROSTŘEDÍ.....	38
2.8.1 Hardwarové požadavky.....	38

2.8.2	Licenční požadavky	38
2.8.3	Softwarové požadavky	39
2.8.4	Konfigurace sítě pro jednotlivé služby	40
2.8.5	Příklad YAML konfigurace portů	40
2.9	STRUKTURA A ARCHITEKTURA SIEM PROSTŘEDÍ	41
2.9.1	Komponenty architektury	41
2.9.2	Topologie SIEM prostředí	42
2.9.3	Grafická reprezentace architektury	42
2.9.4	Konfigurační soubory architektury	44
2.9.5	Příklad konfigurace zapnutých a vypnutých datových zdrojů	44
2.9.6	Příklad definice provisioning parametrů v Terraformu	44
2.9.7	Shrnutí architektury	44
II	PRAKTICKÁ ČÁST	45
3	VÝVOJ A PŘÍPRAVA PROSTŘEDÍ	46
3.1	VÝVOJOVÉ PROSTŘEDÍ A NÁSTROJE	46
3.1.1	Příprava Docker pro běh Dev Containeru	46
3.1.2	Použité nástroje a verze	47
3.1.4	Hierarchie souborů nástroje Packer	48
3.1.5	Jak Packer funguje v praxi	49
3.1.6	Konfigurace kickstart souboru pro automatizovaný build	50
3.1.7	Lokální testování generování obrazu systému (Vagrant box)	51
3.2	TVORBA A NASAZENÍ ZÁKLADNÍHO OBRAZU	52
3.2.1	Automatizovaný deploy pomocí Terraform (cloud-init, guestinfo, disk layout)	53
3.2.2	Ověření připravenosti základního image pro instalaci Splunk	54
4	IMPLEMENTACE SIEM SYSTÉMU	55
4.1	INSTALACE A KONFIGURACE SPLUNK ENTERPRISE 9.3.4	55
4.1.1	Použití Ansible pro instalaci Splunk Enterprise 9.3.4	55
4.1.2	(Plánovaná) instalace Splunk Enterprise Security 8.0	55
4.1.3	Konfigurace základních komponent (porty, služby, spuštění)	56
4.1.4	Automatizace pracovního postupu pomocí Makefile	57
4.1.5	Definice uživatelů a rolí (učitel, student)	57
4.1.6	Import uživatelů z CSV a přiřazení oprávnění	58
4.2	KONFIGURACE DATOVÝCH VSTUPŮ A ZPRACOVÁNÍ	59
4.2.1	Napojení datových zdrojů: Windows logy	59
4.2.2	Napojení datových zdrojů: Linux / Syslog	59
4.2.3	Napojení datových zdrojů: NetFlow/IPFIX	59
4.2.4	Příprava testovacích vzorků (logy, pcap) ve spolupráci s vedoucím	59
4.2.5	Parsování dat ve Splunku – Suricata JSON	60
5	TESTOVÁNÍ ŘEŠENÍ	61
5.1	TEST BĚHU PACKER BUILD	62
5.2	TEST BĚHU TERRAFORM APPLY – SPLUNK UNIVERSAL FORWARDER	63
5.2.1	Screenshots z testu Terraform apply	64

5.3	TEST INSTALACE SPLUNK ENTERPRISE POMOCÍ TERRAFORM.....	65
5.3.1	Detail testu Splunk Enterprise.....	66
ZÁVĚR		67
SEZNAM POUŽITÉ LITERATURY.....		69
SEZNAM OBRÁZKŮ		73
SEZNAM TABULEK.....		74
SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....		75
SEZNAM PŘÍLOH		76

ÚVOD

V současné době je monitorování a analýza bezpečnostních událostí klíčovým prvkem kybernetické bezpečnosti. Organizace čelí rostoucímu množství bezpečnostních hrozeb, a proto je nutné implementovat efektivní nástroje pro detekci a reakci na incidenty. Security Information and Event Management (SIEM) systémy umožňují sběr, analýzu a korelaci bezpečnostních událostí z různých zdrojů v síti, čímž napomáhají rychlé identifikaci podezřelých aktivit a potenciálních útoků.

Tato bakalářská práce se zaměřuje na analýzu možností vytvoření SIEM prostředí pro výuku kybernetické bezpečnosti, specifikaci požadovaných funkcí, výběr vhodných prostředků pro realizaci, návrh prostředí, implementaci v testovací infrastruktuře a ověření funkčnosti navrženého řešení.

Jedním z hlavních problémů při nasazení SIEM řešení je jeho technická složitost a časová náročnost, zejména pokud je nutné opakované nasazení pro výukové účely. Z tohoto důvodu práce klade důraz na automatizaci celého procesu, která zajistí efektivní, konzistentní a opakovatelné nasazení. K dosažení tohoto cíle budou využity technologie jako Terraform, Ansible a VMware, které umožní infrastrukturu definovat jako kód (Infrastructure as Code, IaC) a usnadní její správu a údržbu.

Vytvořené řešení se využije zejména v akademických předmětech:

- **Forenzní vědy** – studenti budou analyzovat bezpečnostní incidenty na reálných datech.
- **Počítačové viry a bezpečnost** – poskytne platformu pro testování detekce hrozeb a analýzu škodlivého kódu.

Práce je rozdělena do několika částí, které reflektují jednotlivé kroky řešení:

- Teoretická část obsahuje:
 - **Analýzu dostupných SIEM řešení** – jejich význam a srovnání různých platforem.
 - **Návrh řešení** – popisuje požadavky na SIEM prostředí, jeho architekturu a výběr vhodných nástrojů pro realizaci.

- Praktická část obsahuje:
 - **Implementaci** – dokumentuje praktickou realizaci a připojení zdrojů dat
 - **Testování** – ověřuje funkčnost celého systému
- **Závěr** shrnuje dosažené výsledky a navrhuje možné budoucí rozšíření.

Hlavním cílem této práce je navrhnout praktické a efektivní řešení pro automatizované nasazení SIEM prostředí, které zajistí rychlé nasazení a snadnou správu. Výsledná SIEM platforma bude primárně sloužit studentům při výuce kybernetické bezpečnosti v konkrétních scénářích.

I TEORETICKÁ ČÁST

1 SECURITY INFORMATION AND EVENT MANAGEMENT

SIEM představuje komplexní soubor technologií navržených k zajištění přehledu a transparentnosti nad celým podnikovým IT systémem. Jeho cílem je poskytovat podporu jak bezpečnostním analytikům, tak i správcům IT. Jak bude ukázáno v následujících kapitolách, SIEM umožňuje nejen tyto základní funkce, ale i pokročilé možnosti analýzy a reakce. SIEM se stal významnou součástí IT infrastruktury v malých, středních i velkých podnicích, v organizacích a ve vládních institucích [1].

1.1 Význam SIEM

SIEM je technologie, která kombinuje správu logů se schopností analýzy bezpečnostních událostí v reálném čase. SIEM pomáhá bezpečnostním týmům identifikovat anomálie, detekovat hrozby a reagovat na bezpečnostní incidenty dříve, než ovlivní chod organizace [2]. Mezi klíčové přínosy [2] relevantní pro potřeby této práce patří:

- Detekce podezřelých aktivit a bezpečnostních hrozeb.
- Monitorování a upozorňování na bezpečnostní události v reálném čase.
- Podpora forenzního vyšetřování a analýzy incidentů.
- Zajištění souladu s bezpečnostními standardy a legislativními požadavky

1.2 Klíčové funkce SIEM

Mezi hlavní funkce SIEM systému podle Millera [1] patří:

- **Log Management** – sběr, normalizace, indexace a archivace logů z různých systémů a aplikací.
- **Zajištění souladu s předpisy (compliance)** – sledování a ověřování shody s bezpečnostními standardy a legislativními požadavky.
- **Korelace událostí** – analýza souvislostí mezi různými bezpečnostními událostmi s cílem identifikovat hrozby.
- **Active response** – možnost automatizované reakce na bezpečnostní incidenty, například blokování škodlivých aktivit.
- **Endpoint security** – monitorování a správa bezpečnostního stavu koncových zařízení v síti.

2 PŘEHLED A SROVNÁNÍ SIEM ŘEŠENÍ

Pro potřeby této práce bylo provedeno srovnání dvanácti aktuálně nejvýznamnějších SIEM řešení. Cílem analýzy je vybrat platformu, která nejlépe odpovídá požadavkům na automatizované nasazení v akademickém prostředí a zároveň poskytuje dostatečnou funkcionalitu pro bezpečnostní cvičení s praktickým využitím v reálné praxi.

Výběr produktů a kritérií hodnocení vychází z uznávaných analytických reportů Gartner Magic Quadrant 2024 [3] a IDC MarketScape 2024 [4], které jsou považovány za standard v oblasti hodnocení bezpečnostních technologií. Tyto reporty jsou široce respektované v odborné komunitě a poskytují objektivní pohled na schopnosti, vyspělost a postavení jednotlivých řešení na trhu.

Výsledné srovnání se zaměřuje nejen na technologické parametry, ale i na praktické aspekty nasazení v podmínkách akademického prostředí, kde je požadována vysoká flexibilita, automatizace a škálovatelnost.

2.1 Výběr produktů pro srovnání

Do srovnání byla zařazena následující SIEM řešení, která patří mezi lídry a vizionáře podle zmíněných analytických reportů [3][4]:

- Splunk Enterprise Security
- Microsoft Sentinel
- IBM QRadar
- Securonix
- Exabeam
- Gurukul
- Google Security Operations (dříve Google Chronicle)
- Devo
- Elastic Security
- OpenText ArcSight
- LogRhythm
- Rapid7 InsightIDR

Výběr zahrnuje spektrum komerčních i open source řešení, aby bylo možné objektivně porovnat jejich vlastnosti a vhodnost pro požadované scénáře.

2.2 Kritéria hodnocení

Výběr následujících kritérií reflektuje jak technologické požadavky moderních bezpečnostních řešení, tak praktické potřeby akademického prostředí, kde je kladen důraz na efektivitu, flexibilitu, otevřenost a snadnou správu prostřednictvím automatizace.

S ohledem na provoz v dlouhodobém produkčním nasazení a na základě aktuálních poznatků z oblasti DevOps a doporučení z odborných zdrojů [5] byly jako doporučené operační systémy vybrány tyto distribuce ve svých posledních stabilních verzích:

- Ubuntu Server LTS (verze 24.04)
- Debian Stable (verze 12 „Bookworm“)
- Red Hat Enterprise Linux 9.5 (i s komunitními deriváty: AlmaLinux, Rocky Linux)

Tyto systémy jsou dle současných standardů považovány za vhodné pro enterprise a akademické nasazení díky své kompatibilitě s moderními kontejnery, nástroji pro CI/CD a se správou infrastruktury jako kód.

Před samotným hodnocením jednotlivých platforem je tedy u každého řešení nejprve ověřena tato klíčová prerekvizita, kterou je podpora pro on-premises nasazení ve výše uvedených distribucích. Ta je zásadní pro flexibilitu a praktickou implementaci. Bez této vlastnosti není možné dané řešení nasadit v prostředí VMware ESXi, které je základem testovací infrastruktury této práce. Pokud řešení tuto podmínku nesplňuje, není dále hodnoceno.

Pro následné hodnocení byla definována kritéria:

1. Pokročilost funkcí SIEM a připravenost na SOAR

Zohledňuje celkové schopnosti pro pokročilou správu a korelace událostí, automatizace reakcí (SOAR), podporu compliance a vyspělost detekčních mechanismů.

2. Enterprise podpora (support, komunita)

Hodnotí vhodnost pro Enterprise nasazení, dostupnost profesionální podpory, velikost uživatelské komunity a dostupnost znalostních zdrojů.

3. Škálovatelnost

Posuzuje možnosti horizontálního a vertikálního škálování v různých prostředích, zejména s ohledem na on-premises nasazení, které je preferováno v akademických infrastrukturách.

4. Dokumentace (dostupnost, kvalita)

Hodnotí rozsah a kvalitu oficiální dokumentace, dostupnost výukových materiálů a podpory pro vývojáře.

5. Automatizace (API, playbooky)

Sleduje možnosti integrace s automatizačními nástroji (např. Ansible), dostupnost API a podporu tvorby automatizačních skriptů a playbooků.

V následující kapitole přejdeme k posouzení vybraných platforem podle těchto kritérií.

2.3 Analýza jednotlivých SIEM řešení

V této kapitole jsou postupně analyzovány jednotlivé SIEM platformy. U každého řešení je nejprve ověřena klíčová prerekvizita.

Na závěr kapitoly bude provedeno bodové ohodnocení kritérií podle stupnice (1–5, kde 1 znamená, že kritérium není splněno vůbec, a 5 znamená jeho plné splnění), následně bude provedeno shrnutí výsledků a vybrána nejvhodnější platforma pro praktickou část práce.

2.3.1 Splunk Enterprise Security

Splunk Enterprise Security (Splunk ES) je nadstavba řešení Splunk Enterprise, které plně podporuje on-premises instalaci na OS Alma Linux 9 [6].

1. Pokročilost funkcí SIEM a připravenost na SOAR

Prémiová aplikace Splunk Enterprise Security 8.0 přináší moderní inovace, včetně integrovaného pracovního prostředí, nativní integrace se Splunk SOAR, pokročilých detekčních mechanismů (např. Finding Groups), nativního automatického verzování korelačních pravidel a standardizace dle Open Cybersecurity Schema Framework (OCSF) [7].

2. Enterprise podpora (support, komunita)

Splunk poskytuje podporu na všech úrovních [8]: placenou podporu přes portál zákaznický Support Portal, komunitní podporu přes Splunk Community, podrobnou dokumentaci pro Splunk ES [9].

3. Škálovatelnost

Splunk plně podporuje jak horizontální škálování (indexer scaling), tak i vertikální škálování (search head scaling) [10].

4. Dokumentace (dostupnost, kvalita)

Splunk má veřejně dostupnou podrobnou dokumentaci pro uživatele (Search Manual), adminy (Admin Manual) i developery (Splunk SDKs) [11].

5. Automatizace (API, playbooky)

Splunk poskytuje open source integraci s Docker prostředím [12]. Poskytuje taky open source playbooky pro Ansible [12].

2.3.2 Microsoft Sentinel

Sentinel je cloud-native a on-premises instalaci neumožňuje, nebude tedy dále hodnocen. [13]

2.3.3 IBM QRadar

QRadar v poslední verzi 7.5 je pro vázaný na zastaralý operační systém Red Hat Enterprise Linux V7.9 64-bit bez plné podpory pro moderní prostředí [14], nebude tedy dále hodnocen.

2.3.4 Securonix

Securonix je v poslední certifikované on-premise verzi SNYPR 6.4 podporovaný pouze na operačních systémech CentOS 7.9 a RHEL 8.8/8.10 [15]. Je tedy bez plné podpory pro moderní prostředí a nebude tedy dále hodnocen.

2.3.5 Exabeam

Dne 17. července 2024 dokončily společnosti Exabeam a LogRhythm své sloučení a vznikl nový subjekt, který si ponechal název Exabeam [16]. Tato platforma pokračuje v rozvoji výhradně jako cloud-native řešení pod označením Exabeam New-Scale SIEM, nebude tedy dále hodnocen.

LogRhythm, jakožto původně samostatná platforma, si i po fúzi zachoval možnost self-hosted provozu a bude vyhodnocen samostatně v kapitole 2.3.11 LogRhythm.

Za zmínku rovněž stojí související produkt LogRhythm NetMon, který podporuje on-premises nasazení na moderních distribucích (např. Rocky Linux 9.4 a vyšších) [17]. Nicméně vzhledem ke své funkční kategorii NMS (Network Monitoring System) nejde o plnohodnotné SIEM řešení, a proto rovněž není předmětem tohoto hodnocení.

2.3.6 Gurukul

I po kontaktování supportu jsem se nedostal k dokumentaci. Nelze tedy vyhodnotit.

2.3.7 Google Security Operations (dříve Google Chronicle)

Google Chronicle, dnes integrovaný do platformy Google Security Operations, představuje moderní SIEM a XDR řešení určené výhradně pro provoz v cloudovém prostředí Google Cloud Platform (GCP). Vzhledem k tomu, že není k dispozici žádná možnost lokálního nasazení, řešení nebude dále hodnoceno [18].

2.3.8 Devo

Devo nabízí svoji platformu v podobě Devo Platform ve třech variantách SaaS SIEM řešení podle rozsahu poskytovaných funkcí [19]:

- Intelligent SIEM Starter
- Intelligent SIEM
- Intelligent SIEM+

Všechny uvedené varianty jsou však nabízeny výhradně jako cloud-native služby. Z tohoto důvodu Devo nebude dále hodnoceno.

2.3.9 Elastic Security

Elastic Security Self-Hosted verze 9.x (Elastic Defend) podporuje on-premises nasazení na požadovaných Linux distribucích. Předpoklad podpory těchto OS je tedy splněn, a lze přistoupit k hodnocení níže uvedených kritérií. [20]

1. Pokročilost funkcí SIEM a připravenost na SOAR

Pokročilé detekční schopnosti s podporou strojového učení a bohatou knihovnou pravidel. Částečně integrovaný SOAR s automatizovanými akcemi a integrací na externí nástroje. Chybí pokročilejší integrovaný orchestrátor playbooků. [21]

2. Enterprise podpora (support, komunita)

Špičková podpora od Elastic s globálním týmem a aktivní komunitou. Dostupnost množství znalostních zdrojů, fór a partnerů zaručuje rychlé řešení problémů a sdílení know-how. [20]

3. Škálovatelnost

Vynikající horizontální i vertikální škálovatelnost díky distribuované architektuře Elasticsearch. Snadné přidávání nodů a zvládání vysokých objemů dat v reálném čase. [22]

4. Dokumentace (dostupnost, kvalita)

Rozsáhlá a kvalitní oficiální dokumentace doplněná množstvím školení a příkladů. Rezervy v přehlednosti mezi verzemi a lokalizaci mohou ztížit správné nasazení nebo konfiguraci řešení, zvláště v on-premise prostředích. Například nejsou odděleny postupy pro cloud a pro on-premise Elastic Stack [23].

5. Automatizace (API, playbooky)

Otevřená platforma s plnou podporou API umožňuje integraci do různých nástrojů (Ansible, Terraform, SOAR). Vestavěné konektory a reakce dovolují vysoký stupeň automatizace incident response. [22] [23]

2.3.10 OpenText ArcSight

ArcSight ESM ve verzi 7.8 oficiálně podporuje on-premises nasazení pouze na vybraných enterprise distribucích Linuxu: konkrétně Red Hat Enterprise Linux (RHEL) 7.9, 8.8, 8.9, 8.10 a 9.2 (s FIPS i bez), SUSE Linux Enterprise Server 15 SP5 a komunitní klon Rocky Linux 8.9, 8.10 a 9.2. V oficiální matici podporovaných systémů chybí Ubuntu Server 24.04 LTS i Debian 12 „Bookworm“. AlmaLinux rovněž není v seznamu zmíněn. [24]

ArcSight ESM tedy nepodporuje on-premises instalaci na Ubuntu 24.04, Debian 12 ani RHEL 9.5, což znemožňuje nasazení v daném prostředí. Z tohoto důvodu nelze řešení dále hodnotit podle zadaných kritérií, neboť nesplňuje základní požadavek.

2.3.11 LogRhythm

LogRhythm verze 7.20.0 podporuje Linux jen na 2 z 9 komponent, a to Data Indexer a Open Collector. Zbylé komponenty (Data Processor, Web Console, ...) kritické pro běh a používání tohoto řešení jsou podporovány pouze na Windows Server. [25]

Z tohoto důvodu nebude toto řešení dále hodnoceno.

2.3.12 Rapid7 InsightIDR

InsightIDR je poskytován jako cloud-native služba. Zákazník má on-prem pouze Kolektor pro sběr logů, jenž odesílá data do cloudu. Není dostupná verze k instalaci IDR nebo SOAR na vlastním Linux serveru. Toto řešení tedy nebude dále hodnoceno. [26]

Následující tabulka poskytuje souhrnné srovnání vybraných SIEM platforem:

Tabulka 1 Srovnání SIEM

SIEM	LINUX	SIEM	ENTERPRISE	SCALING	DOCS	AUTOMATION	SKÓRE	KOMENTÁŘ
SPLUNK ES	✓	5	4	5	4	5	28	Adept k realizaci řešení
MS SENTINEL	✗	-	-	-	-	-	-	Cloud-native
IBM QRADAR	~	-	-	-	-	-	-	Podpora do verze RHEL 7.9
SECURONIX	~	-	-	-	-	-	-	Podpora do verze RHEL 8
EXABEAM	✗	-	-	-	-	-	-	Cloud-native
GURUCUL	~	-	-	-	-	-	-	Nedostupná dokumentace
GOOGLE SECOPS	✗	-	-	-	-	-	-	Cloud-native
DEVO	✗	-	-	-	-	-	-	Cloud-native
ELASTIC SECURITY	✓	4	5	5	4	5	28	Adept k realizaci řešení
OPENTEXT ARCSIGHT	~	-	-	-	-	-	-	Podpora do verze RHEL 9.2
LOGRHYTHM	~	-	-	-	-	-	-	Řešení závislé na Windows
RAPID7 INSIGHTIDR	✗	-	-	-	-	-	-	Cloud-native

Legenda: ✓ - splňuje, ~ - téměř splňuje, ✗ - nesplňuje
(zdroj: vlastní)

Z bodového srovnání vyplývá, že platformy Splunk Enterprise Security a Elastic Security dosáhly shodného téměř maximálního hodnocení. Obě představují silné kandidáty pro praktické nasazení. Podrobnější porovnání těchto dvou řešení bude provedeno v následující kapitole.

2.4 Vybraný SIEM

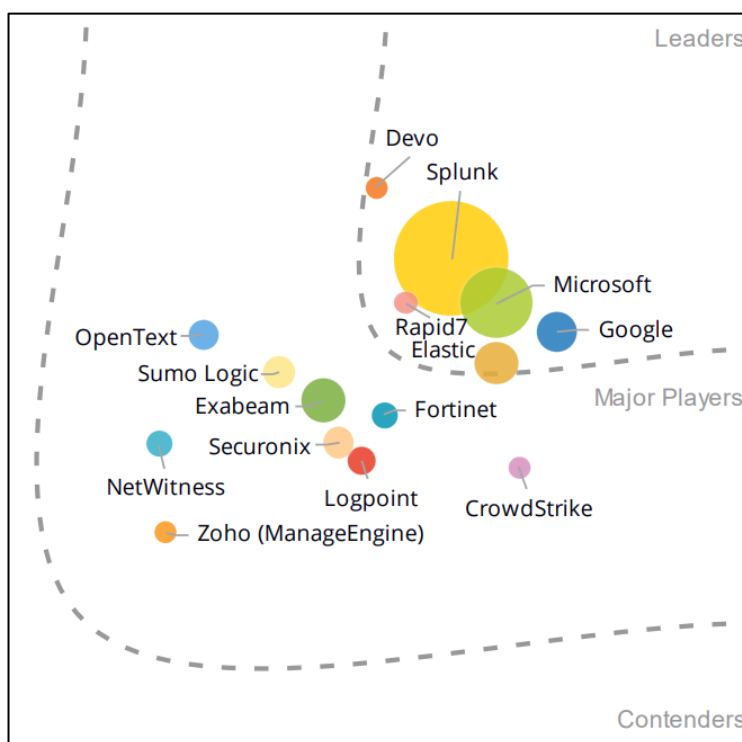
Při bližším porovnání Splunk Enterprise Security a Elastic Security lze říct, že obě platformy jsou kvalitní, každá však cílí na jiný typ nasazení.

Splunk ES je robustní řešení, které je plně integrované, nabízí rozsáhlou funkcionalitu včetně nativní autentizace, správy přístupů, SOAR integrace a centralizované podpory. Díky tomu je vhodný k okamžitému nasazení ve větších organizacích a jeho správa nevyžaduje

velké množství dedikovaných technických prostředků. Nevýhodou jsou vyšší počáteční náklady na licenci a infrastrukturu.

Elastic Security, naopak, staví na open-source přístupu. Je možné jej nasadit bez licenčních poplatků, avšak většina pokročilých funkcí (např. RBAC, strojové učení, auditní logy) je dostupná pouze v placených modulech. Navíc, platforma není tak integrována – skládá se z více samostatných komponent (Elasticsearch, Kibana, Beats, Logstash), jejichž propojení a údržba vyžaduje zkušenější technický personál. V důsledku toho je provoz Elastic Security náročnější z hlediska času a zdrojů.

Pokud do porovnání zahrneme i praktické nasazení a možnosti profesního uplatnění, pak podle analýzy IDC MarketScape 2024 [4] zaměřené na oblast SIEM řešení vychází Splunk jako jednoznačný lídr. Jeho výrazná pozice v segmentu „Leaders“ a velikost bubliny, která znázorňuje tržní podíl, potvrzují jak technologickou vyspělost platformy, tak její rozšíření v praxi. I z tohoto důvodu představuje Splunk nejen technicky vhodné řešení, ale také silný nástroj pro budoucí profesní uplatnění studentů:



Obrázek 1 IDC MarketScape Worldwide SIEM for Enterprise

Zdroj: IDC, 2024 [4]

Na základě výše uvedených výsledků byla zvolena platforma Splunk Enterprise Security jako nejvhodnější řešení pro praktickou realizaci této bakalářské práce.

2.5 Proč automatizace aneb Úvod do DevOps

Po výběru konkrétní SIEM platformy je dalším zásadním krokem rozhodnutí, jakým způsobem bude prostředí technicky nasazeno. Zkušenosti z praxe ukazují, že ruční instalace komplexního systému je časově náročná, náchylná k chybám a obtížně opakovatelná. V rámci této práce, a i v produkčním prostředí je přitom žádoucí, aby bylo možné prostředí snadno vytvořit, obnovit či přenést.

Typickým příkladem manuálního postupu byl tradiční přístup, kdy vývojářský tým dokončil aplikaci a „přehodil ji přes zed“ operačnímu týmu, který měl za úkol ručně provést instalaci a nasazení. Taková praxe často zahrnovala manuální úkony, jako fyzické zapojování serverů či instalaci aplikací a jejich závislostí [27]. Tento způsob může fungovat relativně dobře u malých týmů nebo jednoduchých projektů, ale s růstem společnosti a nárůstem počtu serverů se stává nasazení pomalým, obtížně předvídatelným a náchylným k chybám. Následkem jsou takzvané „snowflake servery“, kde každý server má mírně odlišnou konfiguraci, což vede k jevu známému jako configuration drift a k častějším výpadkům a problémům.

V rámci svých pracovních úkolů jsem se osobně setkal s potřebou rychle nasadit Windows cluster, který bylo nezbytné zprovoznit k vyřešení požadavků zákazníka. Při hledání vhodného řešení jsem narazil na projekt DetectionLab [28]. V průběhu testování se ukázalo, že DetectionLab dokázal za dobu obvyklou pro ruční instalaci jednoho serveru vytvořit a nakonfigurovat celkem tři Windows servery a jeden Ubuntu server. Výsledkem byla plně funkční infrastruktura připravená k okamžitému použití. Tento projekt mě přesvědčil o tom, že automatizovat lze nejen jednotlivé aplikace či konfigurace, ale i celé virtuální servery včetně jejich vzájemných vazeb.

Inspirace DetectionLabem vedla k volbě podobného přístupu i v této práci, kde byla automatizace nasazení SIEM prostředí implementována s využitím technologií, jako jsou Terraform, Ansible, Packer a Vagrant. V následujících podkapitolách budou představeny vybrané principy DevOps a konkrétní nástroje použité pro automatizaci.

2.5.1 Co je to DevOps

Představme si tradiční firmu, která nepoužívala virtualizaci nebo cloudové služby. Pro vývoj softwaru potřebovala fyzický hardware, na němž bylo možné spouštět vývojové nástroje a aplikace. Bylo nutné vybudovat serverovnu, zajistit infrastrukturu v podobě serverových racků, propojení síťových kabelů, elektrického napájení a klimatizace. O tyto činnosti se starala odborná skupina označovaná jako Operations (zkráceně „Ops“).

Druhá skupina, Developers („Devs“), se zabývala výhradně psaním programového kódu. Jakmile dokončili určitý softwarový celek, předali jej týmu Ops, jehož úkolem bylo ručně zajistit nasazení této aplikace na připravenou infrastrukturu. Tým Ops často prováděl instalaci manuálně, například zkopírováním kódu na server a spuštěním potřebných příkazů v příkazové řádce. Mezitím tým vývojářů čekal na potvrzení, že aplikace běží správně v cílovém prostředí. Tento oddělený přístup způsoboval prodlevy při implementaci a nasazování kódu, navíc kvůli ručním úkonům často docházelo k chybám či překlepům, jejichž řešení vyžadovalo další neplánované zásahy, někdy i v noci nebo ve velmi nevyhovující dobu.

V současné době, kdy neustále roste počet bezpečnostních hrozeb a je nutné rychle reagovat vydáváním aktualizací a záplat, si firmy stále více uvědomují neefektivitu ruční správy serverů. Z tohoto důvodu vznikly služby poskytované nadnárodními společnostmi, jako je například Amazon Web Services (AWS), které začaly nabízet hotová datacentra ve formě komerčně dostupných cloudových služeb. Tato datacentra poskytují infrastrukturu jako flexibilní, snadno konfigurovatelnou službu založenou na šablonách a programových požadavcích. Díky tomu se výrazně snížila potřeba fyzické správy serverů ze strany týmu Ops. Namísto toho se členové tohoto týmu začali učit, jak požadovanou infrastrukturu definovat a obstarávat prostřednictvím kódu.

To umožnilo užší spolupráci mezi týmy Dev a Ops, protože oba týmy začaly používat stejný nástroj – programový kód. Právě z tohoto propojení a z potřeby těsnější integrace vývoje a provozu vznikla metodika DevOps [27].

Postupem času se ukázalo jako výhodné zapisovat stále více kroků potřebných ke konfiguraci infrastruktury přímo pomocí kódu, čímž vznikl koncept Infrastructure as Code. Právě tomu se věnuje následující kapitola.

2.6 Infrastruktura jako kód (IaC)

Koncept Infrastructure as Code (IaC) spočívá v tom, že infrastruktura je definována, nasazována, aktualizována i odstraňována pomocí kódu. Místo manuální konfigurace prostředí uživatel píše skripty nebo konfigurační soubory, které určují požadovaný stav celé infrastruktury.

Tento přístup představuje výrazný posun v chápání toho, co znamená „server“. Infrastruktura přestává být fyzickým objektem a stává se programovatelným zdrojem. Pomocí IaC lze popsat celý životní cyklus serveru – od vytvoření výpočetních prostředků, jejich napojení na síť, až po instalaci, konfiguraci aplikací, databází a jejich naplnění daty. [27]

Nástroje pro správu infrastruktury jako kódu (IaC) lze rozdělit do několika hlavních kategorií podle jejich zaměření: [27]

1. Skripty psané jednorázově (Ad hoc scripts)
 - a. Používány pro rychlé řešení specifických úloh bez širší opakovatelnosti (např. Bash, Python)
2. Nástroje pro správu konfigurace (Configuration management tools)
 - a. Zajišťují definici a udržení požadovaného stavu systémů (např. Ansible, Chef, Puppet).
3. Nástroje pro tvorbu šablon serverů (Server templating tools)
 - a. Umožňují definovat standardizované obrazy systémů s přednastavenými vlastnostmi (např. Packer, Docker, Vagrant)
4. Nástroje pro orchestraci (Orchestration tools)
 - a. Koordinují více kroků nebo služeb do automatizovaného, propojeného celku (např. Kubernetes, Docker Swarm, Nomad).
5. Nástroje pro zřízení serveru (Provisioning tools)
 - a. Slouží k vytvoření nových výpočetních prostředků, např. virtuálních strojů nebo kontejnerů (např. Terraform).

2.6.1 Skripty psané jednorázově

Nejpřímějším způsobem, jak automatizovat jednoduchý úkol, je využití ad hoc skriptu. Ten obvykle obsahuje sadu příkazů, proměnné a jednotlivé kroky, které se mají vykonat v daném pořadí. Hodí se zejména v situacích, kdy je třeba rychle provést opakovanou činnost bez potřeby rozsáhlé infrastruktury nebo správy konfigurace. [27]

Modelovým příkladem může být požadavek na více serverech nainstalovat prohlížeč Firefox a otevřít rozhraní iLO karty na adrese <https://192.168.1.100>. Následující Bash skript může celý proces výrazně urychlit:

```
#!/bin/bash
ILO_IP="192.168.1.100"
sudo dnf install -y firefox
firefox "https://$ILO_IP"
```

2.6.2 Nástroje pro správu konfigurace

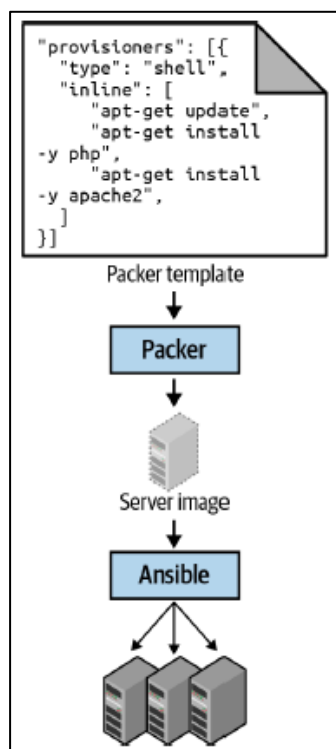
Sem patří nástroje jako Chef, Puppet a Ansible. Ansible má dobře čitelnou syntaxi, nemusí se instalovat agenti a je push based – tedy technik má plnou kontrolu nad tím, kdy se aplikuje konfigurace [28]. Umožňuje automatizovat nastavení serverů pomocí tzv. playbooků. Playbook je soubor ve formátu YAML, který popisuje:

- Na které servery (hosts) se má změna aplikovat,
- Úlohy (tasks), které se na nich mají vykonat, v jakém pořadí.

Například pro náš modelový příklad:

```
- name: Instalace Firefoxu a spuštění iLO rozhraní
  hosts: all
  become: yes
  vars:
    ilo_ip: "192.168.1.100"
  tasks:
    - name: Instalace Firefoxu
      dnf:
        name: firefox
        state: present
```

2.6.3 Nástroje pro tvorbu šablon serverů



Obrázek 2 Packer

Zdroj: Terraform [27]

Alternativou ke klasickému modelu správy konfigurace jsou nástroje pro tvorbu šablon serverů, jako jsou Docker, Packer nebo Vagrant, jejichž popularita v posledních letech výrazně vzrostla. Namísto toho, aby se každý server konfiguroval jednotlivě pomocí skriptů či konfigurací, je cílem těchto nástrojů vytvořit předpřipravený obraz (image) serveru, který obsahuje kompletní operační systém, nainstalovaný software, soubory a další potřebné prvky.

Výsledný image slouží jako plně samostatný a znovupoužitelný otisk prostředí, který lze následně pomocí jiných nástrojů (např. Terraform) nasadit na více serverů současně. Tento přístup výrazně zvyšuje konzistenci mezi jednotlivými instancemi a usnadňuje správu prostředí. [27]

2.6.4 Nástroje pro orchestraci

V rozsáhlejších scénářích, kde běží více kontejnerů nebo virtuálních strojů, nestačí pouze statické šablonování. Je potřeba řešit dostupnost výpočetních zdrojů, efektivní využití výkonu, aktualizace za provozu, monitorování stavu služeb a dynamické škálování podle zátěže. Orchestrace umožňuje tyto aspekty řídit centrálně a automatizovaně.

Typickým nástrojem pro tyto účely je Kubernetes, který umožňuje definovat požadovaný stav aplikací a jejich životní cyklus jako kód. Příkladem může být konfigurace, která udržuje 3 repliky služby a provádí aktualizace formou rolling update: [27]

```
replicas: 3  
strategy:  
  rollingUpdate:  
    maxSurge: 3  
    maxUnavailable: 0  
  type: RollingUpdate
```

2.6.5 Nástroje pro zřízení serveru

Provisioning nástroje slouží k vytváření samotné infrastruktury – tedy výpočetních prostředků, na kterých následně běží aplikace. Na rozdíl od nástrojů pro správu konfigurace či šablonování serverů, které řeší, co se má uvnitř serveru spouštět, provisioning definuje, jaké servery a další komponenty mají vůbec vzniknout.

Pomocí těchto nástrojů lze vytvářet nejen samotné servery, ale také databáze, load balancery, síťová pravidla, certifikáty, monitoring, fronty zpráv či firewall pravidla. Vše je definováno jako kód, a tedy verzovatelné, opakovatelné a automatizovatelné.

V této práci je použit nástroj Terraform, který umožňuje popsat cílovou infrastrukturu v jednoduchém, deklarativním jazyce. Využívá se k nasazení virtuálních strojů na platformě VMware ESXi, kde definuje jejich výpočetní parametry, síťové rozhraní i připojený systémový obraz. Součástí konfigurace může být i skript (např. pomocí `user_data`), který se provede při startu systému.

Provisioning často spolupracuje s nástroji pro tvorbu šablon serverů – typicky se nejprve vytvoří image pomocí Packeru a následně je tento image automatizovaně nasazen pomocí Terraformu. Tato kombinace tvoří základ tzv. neměnné infrastruktury, kde každý server je vytvořen vždy znovu, nikoliv upravován za běhu. [27]

2.7 Automatizace nasazení SIEM

V předchozích kapitolách jsme se věnovali nástrojům, které tvoří základ moderní infrastruktury jako kódu. Nyní je však třeba stanovit konkrétní cíle, které má automatizace v kontextu tohoto projektu splňovat. Teprve na základě těchto požadavků můžeme zodpovědně vybrat vhodné nástroje a navrhnout architekturu celého řešení.

Automatizace v rámci této práce má sloužit především k tomu, aby bylo možné připravit SIEM prostředí pro výukové účely jednoduše, spolehlivě a opakovaně. Současně musí zůstat dostatečně flexibilní, aby bylo možné prostředí upravit pro různé scénáře a úrovně cvičení.

2.7.1 Cíle automatizace

Hlavním cílem je umožnit rychlé, opakovatelné a přizpůsobitelné nasazení SIEM prostředí bez nutnosti složité manuální konfigurace. Toto řešení poskytne učitelům dvě možnosti:

1. **Použití hotového, otestovaného image OVA** – okamžitě funkční prostředí připravené k nasazení.
2. **Možnost úpravy konfigurace a generování nového image** – učitelé mohou v případě potřeby aktualizovat nebo přizpůsobit SIEM prostředí pomocí nástrojů Ansible, Dockerfile, Terraform a Packer.

Tímto způsobem zajistíme maximální flexibilitu, která umožní:

- Rychlé nasazení bez nutnosti hlubších znalostí infrastruktury.
- Možnost přizpůsobení zdrojů dat, pravidel a konfigurací podle potřeby.
- Konzistenci a škálovatelnost výukového prostředí.

2.7.2 Uvažované technologie pro automatizaci

Automatizace bude realizována s využitím následujících nástrojů:

Tabulka 2 Technologie pro automatizace

Technologie	Účel
Packer	Automatizovaná tvorba image pro OVA/VM
Terraform	Provisioning virtuální infrastruktury (VMware, Cloud)
Vagrant	Provisioning on-prem infrastruktury (VirtualBox, Docker)
Ansible	Konfigurace systému, nasazení Splunku, připojení zdrojů
Docker	Kontejnery pro testování a škálování OS a SIEM komponent

(zdroj: vlastní)

Tato kombinace nástrojů umožní snadné a opakovatelné nasazení SIEM prostředí. Uživatelé (např. vyučující) tak mohou prostředí jednoduše nasadit nebo aktualizovat bez nutnosti ruční instalace a složité správy konfigurací.

2.7.3 YAML konfigurace a jejich využití v Ansible

Řada nástrojů pro správu konfigurace a infrastruktury používá k popisu instrukcí formát YAML (YAML Ain't Markup Language). Jedná se o jednoduchý, lidsky čitelný formát navržený právě pro účely konfigurace. Zápis je podobný syntaxi jazyka Python – významnou roli v něm hraje odsazení a logická struktura.

YAML soubory jsou textové soubory, které lze upravovat v jakémkoliv editoru. Pomocí odsazení mezerami se vyjadřuje hierarchie bloků. Mezi základní prvky patří:

- klíč-hodnota (např. name: firefox),
- seznamy (např. - update),
- komentáře (začínající znakem #).

Jak uvádí publikace DevOps [28], soubory obvykle začínají třemi pomlčkami ---, používají odsazení pro strukturu a dají se snadno členit do logických celků. Každý jednotlivý krok (tzv. task) má svůj název a popis činnosti.

Autor této práce vnímá YAML jako technický recept – podobně jako v kuchařce definujeme potřebné suroviny a poté přesný postup přípravy. YAML tímto způsobem umožňuje srozumitelně a přesně definovat, co má systém udělat a v jakém pořadí.

V této práci se YAML využívá například v rámci Ansible playbooků, které mohou:

- **definovat zdroje dat** – které logy budou připojeny
- **nastavit porty pro jednotlivé zdroje** – (např. NetFlow na UDP/2055)
- **určit indexy pro uložení dat** – (např. index=wineventlog)
- **zvolit korelace a detekční pravidla** – (např. brute_force_detection)

Ukázka jednoduché YAML konfigurace [28]:

```
---
- name: Install nginx and php
  hosts: www
  become: yes
  tasks:
    - name: Install nginx
      package:
        name: nginx
        state: present
```

2.7.4 HashiCorp Configuration Language (HCL) v Terraformu

Nástroj Terraform od společnosti HashiCorp je jedním z nejpoužívanějších řešení pro deklarativní správu infrastruktury.

Využívá vlastní konfigurační jazyk HCL (HashiCorp Configuration Language), který umožňuje jednoduše a přehledně popsat cílový stav infrastruktury.

Každá infrastruktura je v Terraformu definována sadou textových souborů s příponou .tf, které společně tvoří tzv. modul. Tyto soubory mohou být rozděleny podle své funkce, a přestože jejich pořadí není důležité, v praxi se dodržuje následující struktura:

- **main.tf:** hlavní soubor, kde jsou definovány zdroje (resource) a případně moduly.
- **variables.tf:** deklarace všech vstupních proměnných, které lze použít v main.tf.
- **terraform.tfvars:** samostatný soubor s přiřazenými hodnotami proměnných.
- **outputs.tf:** výstupy – tedy hodnoty, které Terraform vrátí po úspěšném nasazení (např. IP adresa).
- **providers.tf:** specifikace poskytovatelů (např. VMware, AWS, Azure).
- **backend.tf:** konfigurace, kam se má ukládat tzv. stavový soubor (terraform.tfstate), který sleduje reálný stav infrastruktury.

Terraform je stavově orientovaný – to znamená, že si ukládá informaci o posledním známém stavu infrastruktury do souboru terraform.tfstate. Díky tomu dokáže při každém spuštění porovnat aktuální stav s požadovaným stavem a provést pouze potřebné změny. Tento mechanismus zajišťuje vysokou spolehlivost a predikovatelnost.

Pro samotné nasazení a správu infrastruktury používá Terraform tzv. providery – moduly, které komunikují přes API s jednotlivými službami (např. VMware, AWS, Azure) a zajišťují vytvoření virtuálních strojů, sítí, disků atd. [28]

V této práci je Terraform použit k:

- vytvoření virtuálních strojů na platformě VMware ESXi,
- nastavení síťového rozhraní, diskového prostoru a cloud-init metadat,
- automatizovanému propojení se zlatým image připraveným pomocí nástroje Packer.

Díky jednotnému jazyku HCL je možné kombinovat všechny potřebné konfigurace v rámci jednoho projektu a udržovat přehlednou a verzovatelnou definici celé infrastruktury.

2.7.5 HCL v nástroji Packer

Nástroj Packer od společnosti HashiCorp slouží k automatizované tvorbě systémových obrazů (např. OVA, Vagrant box, QCOW2). Od verze 1.7 podporuje zápis konfigurací v jazyce HCL2 (HashiCorp Configuration Language), který sdílí syntaxi s Terraformem. Tento jazyk je deklarativní a přehledný – konfigurace je rozdělena do bloků `source`, `build`, `provisioner` a volitelně `post-processor`.

Základní struktura HCL konfigurace pro Packer vypadá následovně [29]

```
variable "os_version" {
  type    = string
  default = "9.5"
}

source "vmware-iso" "alma" {
  iso_url      = "http://mirror.almalinux.org/almalinux-
${var.os_version}-x86_64.iso"
  iso_checksum = "sha256:..."
  shutdown_command = "shutdown -P now"
  ...
}

build {
  sources = ["source.vmware-iso.alma"]

  provisioner "shell" {
    inline = [
      "dnf update -y",
      "dnf install -y vim"
    ]
  }

  post-processor "vagrant" {
    output = "alma-${var.os_version}.box"
  }
}
```

V této práci je Packer využit pro přípravu předinstalovaného obrazu systému AlmaLinux, který se následně exportuje jako takzvaný golden image.

Konfigurace umožňuje přidat instalační skripty, uživatelské účty, klíče a další potřebné úpravy v rámci jednoho build bloku. Díky tomu je výstupní obraz konzistentní, opakovatelný a připravený k okamžitému použití v dalších nástrojích jako Terraform nebo Ansible.

2.7.6 Konfigurační soubory .conf ve Splunku a podobnost s .ini

Splunk používá textové konfigurační soubory s příponou .conf, které definují nastavení jednotlivých komponent systému. Tyto soubory se ukládají do specifických adresářů podle kontextu:

- výchozí nastavení: \$SPLUNK_HOME/etc/system/default,
- uživatelské úpravy: \$SPLUNK_HOME/etc/system/local,
- konfigurace aplikací: \$SPLUNK_HOME/etc/apps/.

Úpravy se provádějí tak, že se kopie výchozího souboru přenese do složky local a zde se upraví jednotlivé sekce (tzv. stanza). Každá stanza začíná názvem v hranatých závorkách, například [udp://514], a obsahuje související parametry klíč/hodnota. Tímto způsobem se logicky oddělují jednotlivé konfigurace v rámci jednoho souboru [30]

Ukázka:

```
[my_sourcetype]
SHOULD_LINEMERGE = false
KV_MODE = none
```

Formát konfiguračních souborů Splunku je velmi podobný klasickému INI formátu, který se používá pro jednoduché konfigurační soubory v mnoha systémech. Oba formáty využívají klíče a hodnoty, rozdíl je především v terminologii – Splunk používá označení stanza, zatímco v INI souborech se mluví o sekcích. Tento formát se často objevuje i v souborech pro Ansible inventáře, kde je dobré tyto principy znát [31]

2.7.7 Shrnutí přehledu konfiguračních formátů a nástrojů

V této kapitole byly představeny klíčové nástroje a konfigurační formáty, které budou v praktické části práce využity pro automatizované nasazení SIEM prostředí. Byl objasněn účel formátu YAML a jeho využití v Ansible, princip deklarativních souborů HCL v Packeru a Terraformu, a také struktura .conf souborů ve Splunku.

Díky těmto nástrojům a jazykům je možné celé prostředí sestavit opakovaně, konzistentně a s minimálním manuálním zásahem.

Tímto je zajištěna srozumitelná a udržitelná konfigurace jednotlivých vrstev systému.

Následující kapitola se již zaměřuje na konkrétní požadavky na prostředí, ve kterém bude tato automatizace realizována.

2.8 Požadavky na prostředí

Pro efektivní nasazení SIEM prostředí ve výuce je nutné definovat minimální a doporučené hardwarové a softwarové požadavky. Pro účely této práce bude použito Splunk All-in-One v nejnovější verzi Splunk 9.4 s Enterprise Security 8.0.

2.8.1 Hardwarové požadavky

Požadavky na hardware se liší podle počtu studentů a zpracovávaných dat. Níže jsou uvedeny možné scénáře konfigurace:

Tabulka 3 Hardwarové požadavky

Konfigurace	Počet studentů	CPU	RAM	Disk	Sít'
Minimální	1–6	16	32	150 GB SSD	1 Gbps
Střední	7–12	24	64	300 GB SSD	1 Gbps
Výkonná	13–18	36	92	400 GB SSD	1-10 Gbps

(zdroj: *Minimum specifications for a production deployment [10]*)

Konfiguraci lze upravit v Terraform – main.tf souboru před nasazením.

2.8.2 Licenční požadavky

Navržené řešení využívá základní komponentu Splunk Enterprise, která je dostupná ve zkušební (trial) verzi s plnou funkcí po dobu 60 dní od instalace. Tato varianta umožňuje studentům a školám testovat prostředí bez potřeby komerční licence. [8]

Pro pokročilé analytické funkce a práci s bezpečnostními scénáři je využita nadstavba Splunk Enterprise Security (ES). Tato aplikace není volně dostupná a je poskytována pouze školám zapojeným do programu Splunk Academic Alliance. V rámci této iniciativy je možné získat instalační balíček na základě žádosti přes fakultního zástupce. Základní podmínkou je aktivní studium v oblasti informatiky a členství vysoké školy v akademickém programu. [32]

Pro realizaci této práce je uvažována tato verze Splunk ES:

- Verze: Splunk Enterprise Security 8.0.40
- Soubor: splunk-enterprise-security_8040.spl
- Velikost: 709 922 728 bajtů
- SHA-256:
2041c69a0752a5bb794ba9d164ef6613403752f0208cbfa50b8686021a395783

2.8.3 Softwarové požadavky

Nasazení proběhne na virtuálním stroji v prostředí VMware ESXi 7, přičemž jako hlavní OS bude použit AlmaLinux 9.5.

Tabulka 4 Softwarové požadavky

Komponenta	Podporované verze
Operační systém	AlmaLinux 9.5 x86_64 Minimal
Virtualizace	VMware ESXi 7
Splunk	Splunk Enterprise 9.4.1
Splunk ES	Splunk Enterprise Security 8.0.40
Ansible	ansible [core 2.15.13] python version = 3.9.21 (main, Dec 5 2024, 00:00:00) jinja version = 3.1.6
Terraform	OpenTofu v1.9.1 on linux_amd64
Packer	Packer 1.9.5
Docker	Docker version 28.1.1, build 4eba377327
VS Code	Visual Studio Code Version: 1.100.2 (Universal) + Extension Pack: Remote Development Version 0.26.0 ms-vscode-remote.vscode-remote-extensionpack

(zdroj: vlastní)

Na přípravu obrazu systému a testování nasazení je využito kontejnerizované vývojové prostředí typu Dev Container, které běží v prostředí Docker. Pro komfortní práci s konfiguračními soubory a nástroji jako Terraform, Packer nebo Ansible je doporučeno využít Microsoft Visual Studio Code s rozšířením Dev Containers. [33]

Toto prostředí umožňuje jednoduše sestavit a spustit image systému, testovat konfigurace a ověřovat build procesy izolovaně a bez vlivu na hlavní systém uživatele.

2.8.4 Konfigurace sítě pro jednotlivé služby

Každý SIEM systém musí správně přijímat a zpracovávat data z různých zdrojů. Volitelné služby a porty budou definovány v YAML souborech, aby bylo možné snadno přidat nebo odebrat komponenty.

Tabulka 5 Konfigurace sítě pro jednotlivé služby

Služba	Výchozí port	Protokol	Funkce
Splunk Web UI	8000	TCP	Webové rozhraní pro vyhledávání a administraci
Splunk Indexer	9997	TCP	Receiving port pro data ze Splunk UF – agentů.
Splunk HEC	8088	TCP	HTTP Event Collector (NetFlow, DB Connect JSON a další)
Syslog	514	TCP/UDP	Příjem logů ze switchů a routerů
NetFlow/IPFIX	2055	UDP	Sběr síťového provozu
SSH	22	TCP	Vzdálený přístup, Ansible, konfigurace

(zdroj: vlastní zpracování dle Splunk Docs [8])

2.8.5 Příklad YAML konfigurace portů

```
services:
  splunk_web:
    enabled: true
    port: 8000
    protocol: tcp
  splunk_indexer:
    enabled: true
    port: 9997
    protocol: tcp
  syslog:
    enabled: true
    port: 514
    protocol: udp
```

Tímto byly definovány hardwarové, softwarové a licenční požadavky potřebné pro běh celého SIEM prostředí. V následující kapitole je navržena struktura a architektura celého řešení, která z těchto požadavků vychází a propojuje jednotlivé komponenty do funkčního celku.

2.9 Struktura a architektura SIEM prostředí

Navrhovaná architektura SIEM prostředí je založena na Splunk All-in-One, který bude sloužit jako hlavní analytická platforma pro sběr, indexaci a analýzu bezpečnostních událostí. Architektura umožní snadnou škálovatelnost a flexibilitu díky využití konfigurovatelných souborů a automatizačních nástrojů (Terraform, Ansible, Packer).

2.9.1 Komponenty architektury

Navržené prostředí obsahuje následující klíčové součásti:

- Splunk All-in-One Server
 - a. Hlavní komponenta SIEM, která zajišťuje sběr a analýzu dat.
 - b. (volitelně) Splunk Enterprise Security 8.0 pro pokročilou analýzu hrozeb.
- Uvažované datové zdroje
 - a. Windows – Event Logy (Security, System, Application) z Windows Serverů a pracovních stanic
 - b. Linux – logy z Linuxových serverů a klientských stanic
 - c. NetFlow/IPFIX – Síťový provoz pro analýzu anomálií a detekci útoků
 - d. Splunk Stream
 - e. Suricata
 - f. Syslog
- Automatizace a provisioning
 - a. Packer – generování OVA image pro rychlé nasazení.
 - b. Terraform – provisioning infrastruktury v prostředí VMware ESXi 7+.
 - c. Ansible – konfigurace Splunk prostředí a připojení datových zdrojů.
- Konfigurační management
 - a. Konfigurační soubory umožňují jednoduché přizpůsobení nasazení (např. volba zdrojů dat, síťových portů, bezpečnostních pravidel).

2.9.2 Topologie SIEM prostředí

Navržené SIEM prostředí je rozděleno do tří hlavních vrstev, které společně tvoří plně automatizovatelný, opakovatelný a výukově orientovaný systém:

1. Vývojová a administrátorská vrstva

Tuto vrstvu představuje zařízení učitele nebo správce (typicky notebook), které slouží k přípravě a správě prostředí. Pomocí nástrojů jako Docker, Visual Studio Code a Dev Container je možné v izolovaném prostředí spustit konfigurační skripty a generovat instalační image. Nasazení a konfigurace infrastruktury probíhá s využitím nástrojů Packer, Terraform a Ansible. Klíčové parametry prostředí jsou uloženy ve formátu YAML.

2. Infrastrukturní vrstva

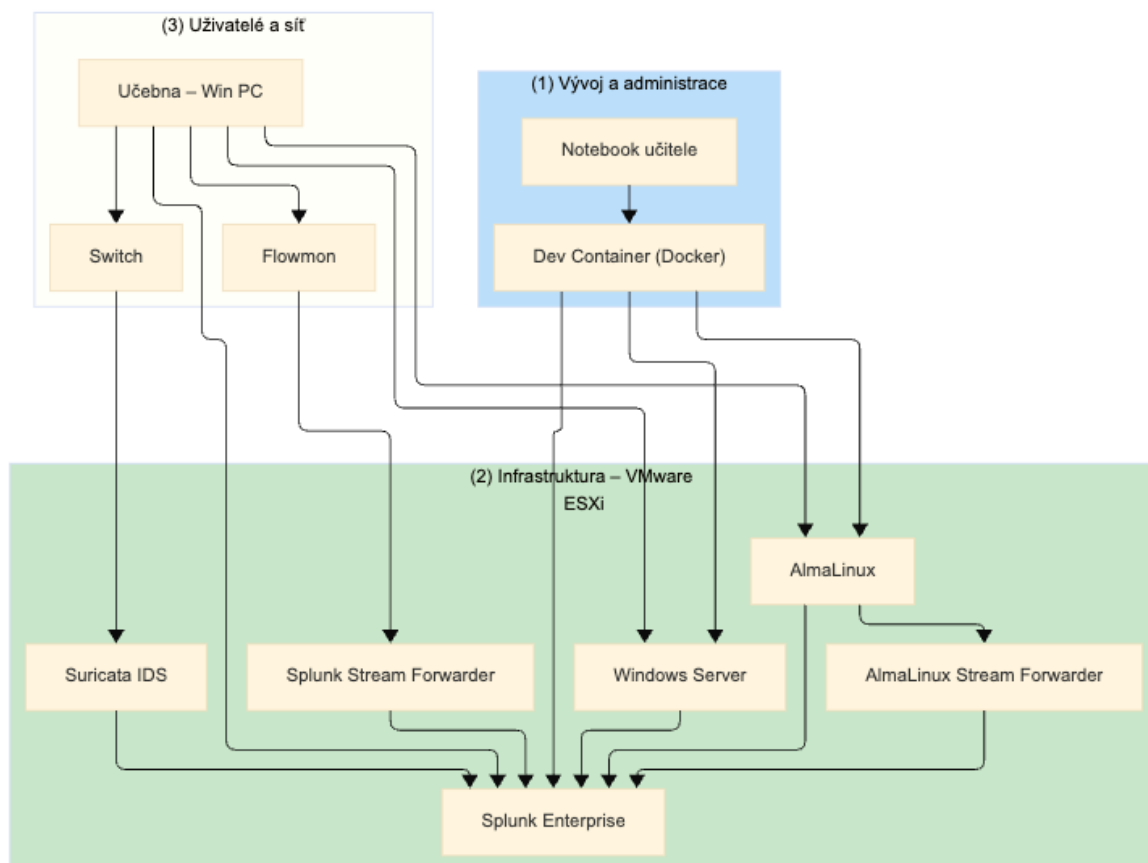
Hlavní komponentou je server s nasazeným Splunk All-in-One, běžící na virtualizační platformě VMware ESXi. V této vrstvě mohou být dále připojeny další virtuální stroje – například klientské stanice se Splunk Universal Forwardery, Linux/Windows systémy generující logy nebo síťové senzory. Tato vrstva tvoří srdce celé SIEM architektury.

3. Uživatelská a síťová vrstva

Studenti a další účastníci výuky pracují s výstupy SIEM prostředí přes webové rozhraní Splunku. Přístup probíhá pomocí běžného webového prohlížeče. Zároveň jsou v této vrstvě generována data pro testování prostřednictvím simulací incidentů, případně skrze reálné zařízení připojená do školní sítě.

2.9.3 Grafická reprezentace architektury

Topologie SIEM prostředí byla znázorněna pomocí jazyka Mermaid, který umožňuje deklarativní generování diagramů. Tento způsob byl zvolen z důvodu jednoduchosti údržby a snadné aktualizace architektury.



Obrázek 3 Architektura SIEM prostředí rozdělená do tří hlavních vrstev.

(zdroj: vlastní tvorba pomocí mermaidchart.com) [34]

Diagram znázorňuje třívrstvou architekturu navrženého SIEM prostředí:

- **Vrstva 1 – Vývoj a administrace:** Učitel nebo správce spravuje infrastrukturu pomocí Dev Containeru běžícího na svém notebooku. V tomto prostředí jsou připravovány konfigurační skripty (Terraform, Ansible, Packer), které zajišťují plně automatizovaný deploy.
- **Vrstva 2 – Infrastruktura (VMware ESXi):** Obsahuje hlavní komponenty jako je Splunk Enterprise, doplňkové forwardery (Splunk Stream), bezpečnostní senzory (Suricata) a klientské systémy (Windows Server, AlmaLinux). Všechny komponenty přispívají data do Splunku.
- **Vrstva 3 – Uživatelé a síť:** Studenti přistupují k SIEM prostředí přes webové rozhraní. Síťová zařízení (Flowmon, Switch) a stanice z učebny generují data, která jsou sbírána a analyzována v rámci výuky.

2.9.4 Konfigurační soubory architektury

Aby bylo možné snadno upravit architekturu nasazení, budou klíčové parametry definovány v YAML souborech.

2.9.5 Příklad konfigurace zapnutých a vypnutých datových zdrojů

```
data_sources:
  syslog:
    enabled: true
    port: 514
  netflow:
    enabled: true
    port: 2055
```

2.9.6 Příklad definice provisioning parametrů v Terraformu

```
infrastructure:
  vmware:
    cpu: 24
    ram: 64 GB
    disk: 300 GB
  network:
    vlan: 20
    ip: "192.168.20.50"
```

2.9.7 Shrnutí architektury

- Modulární architektura s možností rozšíření.
- Kombinace host-based a síťových zdrojů dat.
- Automatizované nasazení pomocí Packer, Terraform, Ansible.
- Plná konfigurovatelnost pomocí souborů.

II PRAKTICKÁ ČÁST

3 VÝVOJ A PŘÍPRAVA PROSTŘEDÍ

Na základě poznatků z teoretické části bylo navrženo řešení, které umožňuje plně automatizované a opakovatelné nasazení SIEM prostředí. Klíčovým předpokladem takového přístupu je mít připravené vývojové prostředí, ve kterém lze bezpečně testovat, generovat image a ověřovat konfigurace bez zásahu do produkční infrastruktury.

Z tohoto důvodu byl zvolen Dev Container běžící v Dockeru jako základní prostředí pro vývoj a tvorbu infrastrukturních image pomocí nástroje Packer. V následujících podkapitolách je popsán způsob vytvoření a použití tohoto prostředí.

3.1 Vývojové prostředí a nástroje

Pro vývoj, testování a tvorbu image byl využit přístup založený na Dev Container architektuře, která umožňuje spouštění nástrojů jako Packer, Terraform či Ansible v izolovaném a reprodukovatelném prostředí. Tento přístup byl zvolen z důvodu jednoduchosti, přenositelnosti a minimalizace závislostí na lokálním systému.

Vývojové prostředí běželo v Docker kontejneru na vzdáleném virtualizovaném serveru s architekturou amd64, ke kterému bylo přistupováno z primárního zařízení autora (Apple MacBook s ARM64) pomocí rozšíření Remote Development ve Visual Studio Code. Díky tomu bylo možné vývoj realizovat i z platformy, která není plně kompatibilní s cílovým prostředím ESXi (amd64).

Tato architektura umožnila vytvářet buildy image nezávisle na lokálním systému a poskytla moderní a flexibilní prostředí s vývojářským komfortem.

3.1.1 Příprava Docker pro běh Dev Containeru

Jako základ pro vývoj byl zvolen Arch Linux, který díky minimalistické filozofii a aktuálním balíčkům poskytl čisté prostředí pro sestavení vývojového stacku.

Vývojový kontejner běžel na vzdáleném serveru s podporou Dockeru, který umožnil přímé spuštění všech nástrojů potřebných pro přípravu infrastruktury, zejména:

- Packer
- Terraform
- Ansible
- Python s knihovnou jinja2

3.1.2 Použité nástroje a verze

- **Hostitelský systém (remote):**
 - a. OS: Virtualizovaný Linux server (amd64)
 - b. Docker: 28.0.0, build f9ced58158
 - c. Visual Studio Code Extensions:
 - i. Remote Development (Extension Pack)
 - ii. Docker: ms-azuretools.vscode-docker verze 1.29.5
 - iii. Docker DX: docker.docker verze 0.4.9
- **Lokální zařízení (notebook autora):**
 - a. Visual Studio Code: 1.99.3 (Universal)
 - b. Rozšíření:
 - i. Dev Containers: ms-vscode-remote.remote-containers verze 0.409.0
 - ii. Remote – SSH: ms-vscode-remote.remote-ssh verze 0.119.0

3.1.3 První spuštění dev kontejneru

Pro vytvoření vývojového prostředí stačí v editoru Visual Studio Code otevřít složku obsahující předpřipravenou skrytou složku **.devcontainer** (s tečkou na začátku).

Tato složka obsahuje:

- **Dockerfile** – definuje výchozí obraz založený na AlmaLinuxu, instaluje nástroje jako ansible, packer, opentofu, python3, ssh server, a utility jako wget nebo unzip.
- **files/VMware-ovftool** – přibalený instalační balíček pro nástroj ovftool sloužící k práci s VMware image (import/export).
- **devcontainer.json** – konfigurační soubor určující název Dev Containeru, cestu k Dockerfile a propojení aktuální pracovní složky jako /workspace uvnitř kontejneru.

Pokud Visual Studio Code automaticky nenabídne spuštění prostředí, lze ho ručně inicializovat přes příkaz „**Dev Containers: Reopen in Container**“. Pro návrat zpět do systému s Docker Daemonem lze využít příkaz „**Dev Containers: Reopen Folder in SSH**“.

První inicializace může trvat několik minut v závislosti na rychlosti připojení. Úspěšné spuštění poznáme podle a podle dostupného terminálu s 64bitovým jádrem a stavového řádku, kde se zobrazí název prostředí ve formátu:

Dev Container: Splunk Lab Toolbox @ (umístění, např. IP adresa)

3.1.4 Hierarchie souborů nástroje Packer

Tímto je vývojové prostředí úspěšně připraveno a obsahuje vše potřebné pro činnost nástroje Packer. Měli bysme ve workspace vidět tuto stromovou strukturu. Položky označené **(tmp)** vznikají až činností Packeru. Položky označené **(custom)** jsou upravené pro potřeby práce. A označení číslem znamená, že jsou převzaty z cizího zdroje. Může být i kombinace:

```

packer/
├── ansible/                                # Konfigurace pro provisioner
│   ├── roles/                             # Jednotlivé Ansible role
│   │   ├── cleanup_vm/                   # Vyčištění image [34]
│   │   ├── cloudinit_support/           # Podpora cloud-init (custom)
│   │   ├── unified_boot/                # Nastavení UEFI bootování [34]
│   │   ├── vagrant_guest/               # Integrace s Vagrantem [34]
│   │   └── vmware_guest/                 # Spec. nastavení VMware [34]
│   └── vagrant.yml                       # Hlavní Ansible playbook [34]
├── box_contents/                          # Výstupní soubory z buildu
│   ├── AlmaLinux-*.vmdk                  # Diskový obraz virtuálního (tmp)
│   ├── AlmaLinux-*.nvram                 # Konfigurační soubor NVRAM (tmp)
│   ├── AlmaLinux-*.mf                    # Manifest soubor (tmp)
│   ├── AlmaLinux-*.ovf                   # Popis image v OVF (tmp)
│   ├── metadata.json                     # Metadata boxu pro Vagrant (tmp)
│   └── Vagrantfile                       # Konfig. boxu pro Vagrant (tmp)
├── http/                                  # Server složka kickstart
│   └── almalinux-9.vag*.ks                # Kickstart soubor (custom)
├── AlmaLinux-9-Vag*.box                   # Výstupní box pro Vagrant (tmp)
├── AlmaLinux-9-Vag*.ova                   # Exportovaný obraz OVA. (tmp)
├── almalinux-template.json.pkr.hcl        # Packer šablona [34] (custom)
├── variables.pkr.hcl                     # Výchozí proměnné [34] (custom)
└── versions.pkr.hcl                      # Verze pluginů [27] (custom)

```

3.1.5 Jak Packer funguje v praxi

Na základě připravené struktury adresáře packer/ je možné spustit proces tvorby tzv. zlatého obrazu (Golden Image) systému AlmaLinux. Veškeré konfigurační kroky jsou řízeny pomocí souborů ve formátu HCL, přičemž hlavní šablonou pro sestavení packer image je soubor `almalinux-template.json.pkr.hcl`, ve kterém je definována sekce `build`.

Po spuštění Packeru dojde k automatickému načtení definic bloků `source`, `build` a `provisioner`. V části `source` je specifikován způsob vytvoření nové virtuální instance na platformě VMware ESXi, včetně odkazu ke stažení instalačního ISO obrazu AlmaLinux, který je následně připojen jako zdroj pro inicializaci.

Aby bylo možné provést plně automatizovanou instalaci systému, využívá se parametr `boot_command`, který skrze simulaci klávesových vstupů do zavaděče (GRUB) přesměruje instalaci na kickstart skript (`.ks`). Ten je uložen a zpřístupněn z lokálního HTTP serveru, který je spuštěn v rámci Dev Containeru. Přístupnost z prostředí ESXi je zajištěna pomocí volby `"runArgs": ["--network", "host"]` v `devcontainer.json`.

Kickstart skript provede například následující kroky:

- vytvoření diskových oddílů (`/`, `/boot`, `swap`) pomocí `parted`,
- vytvoření uživatelských účtů, nastavení hesel a oprávnění,
- aktivaci přihlášení jako `root` (`failsafe`),
- instalace a konfigurace základních balíčků a utilit (např. `wget`, `sudo`),
- základní systémové úpravy pro potřeby Splunk (např. vypnutí `selinux`)

Po instalaci přechází Packer do fáze `provisioner`, kde jsou pomocí Ansible rolí provedeny další konfigurační kroky (např. finální úpravy, čištění image, vypnutí služeb, podpora `cloud-init`). V závěrečné fázi `post-processor` je výstupní obraz exportován do formátu Vagrant `.box` nebo `.ova`, určeného pro nasazení ve školní infrastruktuře.

Pro úspěšné dokončení buildu je nezbytné, aby měl VMware ESXi přístup k web serveru běžícímu v Dev Containeru (typicky na notebooku učitele nebo build serveru).

Tímto je vytvořen image, který lze dále distribuovat a automatizovaně nasazovat pomocí nástroje Terraform.

3.1.6 Konfigurace kickstart souboru pro automatizovaný build

Kickstart soubor umožňuje předat instrukce pro linux anacondu a tím převezme

Zde se nastaví instalační zdroj online repositář a nastavení jako jazyk, časová zóna:

```
url --url https://ftp.sh.cvut.cz/almalinux/9.5/BaseOS/x86_64/os/
text
lang en_US.UTF-8
keyboard us
timezone Europe/Prague --utc
selinux --permissive
firewall --disabled
services --enabled=sshd
```

Zde se rozdělí disk a nastaví generický uživatel vagrant:

```
%pre --erroronfail
parted -s -a optimal /dev/sda -- mklabel gpt
parted -s -a optimal /dev/sda -- mkpart biosboot 1MiB 2MiB set 1
  bios_grub on
parted -s -a optimal /dev/sda -- mkpart '"EFI System Partition"' fat32
  2MiB 202MiB set 2 esp on
parted -s -a optimal /dev/sda -- mkpart boot xfs 202MiB 1226MiB
parted -s -a optimal /dev/sda -- mkpart root xfs 1226MiB 5226MiB
parted -s -a optimal /dev/sda -- mkpart swap linux-swaps 5226MiB 6226MiB
parted -s -a optimal /dev/sda -- mkpart optsplunk xfs 6226MiB 100%
%end

part biosboot --fstype=biosboot --onpart=sda1
part /boot/efi --fstype=efi --onpart=sda2
part /boot --fstype=xfs --onpart=sda3
part / --fstype=xfs --onpart=sda4
part swap --fstype=swap --onpart=sda5
part /opt/splunk --fstype=xfs --onpart=sda6

rootpw vagrant
user --name=vagrant --plaintext --password vagrant
reboot -eject
```

3.1.7 Lokální testování generování obrazu systému (Vagrant box)

Pomocí Packer se vygeneroval základní obraz systému AlmaLinux-9-Vagrant-vmware-9.5-20250426.x86_64.box s velikostí 526 MB. To odpovídá efektivnosti nastavení buildu, které se blíží oficiálnímu Alma Linux boxu na stránce <https://portal.cloud.hashicorp.com/vagrant/discover/almaLinux/9> kde má 466 MB. Větší velikost je způsobena rozšířením linuxu a prerekvizity pro následný deployment přes Terraform a pro chod Splunku.

Tento box se následně rozbálí (vznikne OVF soubor) a převede pomocí OVF tool na OVA image – ten má 532 MB a je připraven na automatický deployment.

3.2 Tvorba a nasazení základního obrazu

Tato část popisuje kompletní proces nasazení základního obrazu systému AlmaLinux 9.5. Pomocí nástroje Packer byl připraven tzv. zlatý obraz (Golden Image), který slouží jako výchozí základ pro všechny další virtuální stroje používané v SIEM prostředí. Obraz je sestaven z minimální instalace AlmaLinuxu a doplněn o přednastavené uživatele, diskové oddíly a balíčky nezbytné pro následné nasazení prostřednictvím nástroje Terraform.

Po úspěšném sestavení je výstupní soubor ve formátu .box převeden pomocí nástroje ovftool do formátu .ova a přesunut do složky terraform. Tento formát je kompatibilní s prostředím VMware ESXi a umožňuje snadný import virtuálního stroje. Funkčnost výsledného obrazu byla ověřena testovacím nasazením v prostředí ESXi 7.0 Update 2, kde bylo potvrzeno, že vytvořená virtuální instance spouští systém korektně a je připravena pro další konfiguraci pomocí provisioningu. Níže je proces popsán v Makefile:

```
# Název boxu a cesta
STAMP=20250426

BOX_NAME=AlmaLinux-9-Vagrant-vmware-9.5-$(STAMP).x86_64
BOX_FILE=$(BOX_NAME).box
OVA_NAME=AlmaLinux-9-Vagrant-VMware-9.5-$(STAMP).x86_64.ova
OVF_NAME=AlmaLinux-9-Vagrant-VMware-9.5-$(STAMP).x86_64.ovf

# Cílový výstup
VM_TARGET=../terraform/VMs/alma-base.ova

# EXTRAKCE BOXU
ova_extract:
    @echo "Rozbaluji .box soubor..."
    cd packer && mkdir -p box_contents
    cd packer && tar xvf $(BOX_FILE) -C box_contents

# PŘEVOD NA OVA
ova_convert:
    @echo "Konvertuji OVF na OVA..."
    cd packer && ovftool ./box_contents/$(OVF_NAME) $(OVA_NAME)
    cd packer && cp ./$(OVA_NAME) $(VM_TARGET)
```

Výsledný „alma-base.ova“ soubor tak tvoří jednotný základ, který je možno dále využít v automatizačním procesu pomocí Terraformu, případně ručně importovat pro rychlé nasazení testovacího nebo produkčního prostředí.

3.2.1 Automatizovaný deploy pomocí Terraform (cloud-init, guestinfo, disk layout)

Pro správu infrastruktury využíváme nástroj OpenTofu (Terraform fork) verze v1.9.1 s poskytovatelem esxi_guest od uživatele josenk [35]. Deploy VM probíhá s využitím předpřipraveného .ova souboru a pomocí klíčových parametrů jako CPU, RAM, rozvržení disků a síťových adaptérů. Zároveň nastavuje uživatele cloud-user i s SSH klíčem.

```
# AlmaLinux 9.5 VM cloned from pre-prepared base VM
resource "esxi_guest" "alma_vm" {
  guest_name      = local.hostname
  disk_store      = var.esxi_datastore
  boot_disk_type  = "thin"
  boot_disk_size  = "40"
  boot_firmware   = "efi"
  memsize         = "8192"
  numvcpus        = "8"
  power           = "on"
  ovf_source      = "./VMs/alma-base.ova"
  network_interfaces {
    virtual_network = var.vm_network
    nic_type        = "vmxnet3"
  }
  network_interfaces {
    virtual_network = "MGMT"
    nic_type        = "vmxnet3"
  }
  network_interfaces {
    virtual_network = "SENSOR"
    nic_type        = "vmxnet3"
  }
  # Zde se předává např. ssh klíč - cloud-init dostane data přes GuestInfo
  guestinfo = {
    "userdata.encoding" = "gzip+base64"
    "userdata"
    base64gzip(data.template_file.cloudinit.rendered)
  }

  guest_startup_timeout = 45
  guest_shutdown_timeout = 30
}
```

Následně se spustí Ansible nad nově vzniklým hostem ve VMware, počká se na SSH spojení, zvětší se disk a nainstaluje se Ansible role `ansible/install-splunk-uf.yml` [36] pro test deploymentu použitím pouze agenta Splunk Universal Forwarder:

```
resource "null_resource" "run_ansible" {
  depends_on = [null_resource.wait_for_ssh]

  triggers = {
    always_run = timestamp()
  }

  provisioner "local-exec" {
    command = <<EOT
      echo "[all]" > ansible/inventory.ini
      echo "alma ansible_host=${esxi_guest.alma_vm.ip_address}"
      ansible_user=cloud-user
      ansible_ssh_private_key_file=cloudinit/id_rsa"      >>
      ansible/inventory.ini

      ANSIBLE_HOST_KEY_CHECKING=False \
        ansible-playbook -i ansible/inventory.ini ansible/disk-
        repartition.yml

      ANSIBLE_HOST_KEY_CHECKING=False \
        ansible-playbook -i ansible/inventory.ini ansible/install-
        splunk-uf.yml
    EOT
  }
}
```

3.2.2 Ověření připravenosti základního image pro instalaci Splunk

Výsledný image má nastavené parametry CPU, paměti, velikost a rozložení disku.

4 IMPLEMENTACE SIEM SYSTÉMU

Tato kapitola popisuje praktickou implementaci automatizovaného SIEM řešení na základě připravené infrastruktury, vytvořeného obrazu systému a konfiguračních šablon. Celý proces je navržen tak, aby byl opakovatelný, přehledný a spravovatelný i bez hluboké znalosti jednotlivých komponent. Využívá nástroje Packer, Terraform (OpenTofu) a Ansible v návaznosti na YAML konfiguraci.

4.1 Instalace a konfigurace Splunk Enterprise 9.3.4

Součástí řešení je možnost nasazení plné verze Splunk Enterprise. Kvůli kompatibilitě se staršími CPU architekturami jsem zvolil nakonec verzi 9.3.4. K tomu slouží Ansible playbook **install-splunk-enterprise.yml**, který se spouští v rámci „make terraform“:

- stáhne instalační archiv z oficiálních stránek Splunk,
- vytvoří systémového uživatele `splunk`,
- rozbalí a nainstaluje software do `/opt/splunk`,
- předá přednastavené heslo admina,
- spustí Splunk jako službu.

Díky parametrizaci je možné tuto instalaci dále rozšiřovat nebo replikovat.

4.1.1 Použití Ansible pro instalaci Splunk Enterprise 9.3.4

Příklad pro boot componentu:

```
- name: Enable boot-start
  command: "{{ splunk_home }}/bin/splunk enable boot-start --
    accept-license --answer-yes -systemd-managed 1 -systemd-unit-file-
    name splunk -create-polkit-rules 1 -user {{ splunk_user }} -group {{
    splunk_group }}"
  become: true
  args:
    creates: "/etc/systemd/system/splunk.service"
```

4.1.2 (Plánovaná) instalace Splunk Enterprise Security 8.0

Tato instalace z důvodu licenčních podmínek je potřeba udělat manuálním nahráním pomocí Splunk Web GUI.

4.1.3 Konfigurace základních komponent (porty, služby, spuštění)

Po minimalizaci image pro vagrant box je potřeba image rozšířit pro výsledný virtuál, nastavil jsem playbook, který zvětší diskové oddíly na požadovanou velikost.

disk-repartition.yml:

```
- hosts: all
  become: true
  tasks:
...
    - name: Delete old swap partition
      command: parted /dev/sda --script rm 5
      ignore_errors: true

    - name: Delete old /opt/splunk partition
      command: parted /dev/sda --script rm 6
      ignore_errors: true

    - name: Resize root partition to 11GB
      shell: |
        printf "Yes\n" | parted ---pretend-input-tty /dev/sda
        resizepart 4 11GB
      args:
        executable: /bin/bash

    - name: Grow root filesystem
      command: xfs_growfs /

    - name: Create new swap partition (4GB)
      command: parted /dev/sda --script mkpart swap linux-swap 11GB
15GB

    - name: Create new /opt/splunk partition (remaining space)
      command: parted /dev/sda --script mkpart optsplunk xfs 15GB 100%

    - name: Format new swap partition
      command: mkswap /dev/sda5

    - name: Format new /opt/splunk partition
      command: mkfs.xfs -f /dev/sda6
```

4.1.4 Automatizace pracovního postupu pomocí Makefile

Pro usnadnění práce během vývoje i nasazení byl vytvořen skript Makefile, který sjednocuje všechny klíčové kroky do přehledného a opakovatelného sledu příkazů. Skript umožňuje pomocí jednoduchých příkazů provádět celý proces – od sestavení obrazu až po deploy a úklid prostředí.

Výhody použití Makefile:

- výrazné zjednodušení práce pro studenta nebo správce,
- jasná struktura jednotlivých kroků a jejich opakovatelnost,
- minimalizace chyb způsobených manuálním zadáváním příkazů,
- možnost rychlé obnovy prostředí při testování nebo výuce.

Přehled dostupných příkazů:

- **make packer** Inicializace a spuštění buildu obrazu pomocí Packer.
- **make ova_extract** Extrakce výstupního .box souboru do struktury .ovf.
- **make ova_convert** Konverze do formátu .ova a přesun do složky Terraform.
- **make terraform** Nasazení připraveného obrazu pomocí nástroje OpenTofu.
- **make users** Dodatečné vytvoření uživatelů pomocí provideru Splunk.
- **make destroy** Vyčištění prostředí – odstranění virtuálních strojů.

Tento skript je navržen tak, aby byl snadno rozšiřitelný o další kroky (například testování nebo exporty), a je plně kompatibilní s vývojovým prostředím v Dev Containeru.

4.1.5 Definice uživatelů a rolí (učitel, student)

Nechal jsem původní definice rolí:

User – může vyhledávat

Power – může navíc tvořit objekty

Admin – může spravovat celý splunk

4.1.6 Import uživatelů z CSV a přiřazení oprávnění

Řešení umožňuje import uživatele, je na to oddělený terrafor, viz Makefile:

```
# Users - krok 5, az bezi Splunk
users:
    cd users && tofu init
    cd users && tofu plan
    cd users && tofu apply -auto-approve
```

Terraform kod, který načítá uživatele z CSV users.csv a aplikuje je na Splunk z Ansible buildu:

```
# main.tf - Terraform config for standalone VMware ESXi using esxi provider
locals {
    splunk_config = yamldecode(file("../terraform/ansible/group_vars/all.yml"))
}
provider "splunk" {
    url
    "${local.splunk_config.bind_ip}:${local.splunk_config.splunk_management_port}" =
    username = "${local.splunk_config.splunk_admin}"
    password = "${local.splunk_config.splunk_password}"
    insecure_skip_verify = true
}
locals {
    asset_csv = csvdecode(file("${path.module}/${var.csv_file_path}"))
}
variable "csv_file_path" {
    type        = string
    description = "Path to the CSV file with user data"
    default     = "users.csv"
}
resource "splunk_authentication_users" "student" {

    name      = "student"
    password  = "student123"
    roles     = ["user"]
    email     = "student@ptlab.dev"
}
resource "splunk_authentication_users" "users" {

    for_each   = { for user in local.asset_csv : user.name => user }
    name       = each.value.name
    email      = each.value.email
    password   = each.value.password
    force_change_pass = false
    roles      = [each.value.role]
```

4.2 Konfigurace datových vstupů a zpracování

4.2.1 Napojení datových zdrojů: Windows logy

Napojení proběhlo v testovací implementaci pomocí addonu Splunk Add-on for Microsoft Windows, je to možnost rozšíření této práce.

4.2.2 Napojení datových zdrojů: Linux / Syslog

Napojení proběhlo v testovací implementaci pomocí addonu Splunk Add-on for Unix and Linux, je to možnost rozšíření této práce.

4.2.3 Napojení datových zdrojů: NetFlow/IPFIX

Napojení proběhlo v testovací implementaci pomocí addonu Splunk Stream, je to možnost rozšíření této práce.

4.2.4 Příprava testovacích vzorků (logy, pcap) ve spolupráci s vedoucím

Jako první vzorek jsme vybrali PCAP files se skenováním SQL Injection a NMAP.

Pro zpracování bude použita Suricata (version 7.0.10 RELEASE) a její výstup do souboru **eve.json**.

A pro parsing bude použita aplikace **Stamus Networks App for Splunk**
<https://splunkbase.splunk.com/app/5262>

odzkousene zpracování:

```
[root@alma-c0615 ~]# suricata -c /etc/suricata/suricata.yaml -r
nmap+zenmap+dirb.pcapng -v
[root@alma-c0615 ~]# sudo suricata-update list-enabled-sources
27/4/2025 -- 23:23:20 - <Info> -- Using data-directory
/var/lib/suricata.
27/4/2025 -- 23:23:20 - <Info> -- Using Suricata configuration
/etc/suricata/suricata.yaml
27/4/2025 -- 23:23:20 - <Info> -- Using /usr/share/suricata/rules for
Suricata provided rules.
27/4/2025 -- 23:23:20 - <Info> -- Found Suricata version 7.0.10 at
/sbin/suricata.
Enabled sources:
- et/open
- aleksibovellan/nmap
- tgreen/hunting
```

4.2.5 Parsování dat ve Splunku – Suricata JSON

Vzorky PCAP byly v testovací implementaci úspěšně zpracované pomocí JSON.

Ukázkový výstup v grafickém rozhraní (viz obrázek níže) potvrzuje správné parsování jednotlivých klíčových polí, jako jsou:

src_ip, dest_ip, proto, signature, event_type, timestamp, category,

a navíc sourcetype=suricata:alert a host dle detekční sondy.

Takto zpracovaná data je možné dále korelovat a zobrazovat v analytických náhledech v rámci Security Essentials, nebo využít jako vstup pro navazující detekční pravidla.

i	Time	Event
>	4/22/25 11:59:51.571 AM	<pre> { [-] alert: { [-] action: allowed category: Attempted Information Leak gid: 1 rev: 2 severity: 2 signature: POSSBL PORT SCAN (NMAP -sS) signature_id: 3400002 } dest_ip: 192.168.213.132 dest_port: 1091 direction: to_server event_type: alert flow: { [-] bytes_toclient: 0 bytes_toserver: 58 dest_ip: 192.168.213.132 dest_port: 1091 pkts_toclient: 0 pkts_toserver: 1 src_ip: 192.168.213.131 src_port: 53 start: 2025-04-22T13:59:51.571874+0200 } flow_id: 2174707319238737 pcap_cnt: 134208 pkt_src: wire/pcap proto: TCP src_ip: 192.168.213.131 src_port: 53 timestamp: 2025-04-22T13:59:51.571874+0200 } </pre> Show as raw text host = alma-c0615.sands.dev source = /tmp/eve.json sourcetype = suricata:alert

Obrázek 4 Suricata - Parsing JSON

Zdroj: Splunk Testovací Nasazení v PTLab

5 TESTOVÁNÍ ŘEŠENÍ

Testování celého řešení probíhalo iterativně v rámci vývojového prostředí založeného na Dev Containeru. Pomocí tohoto prostředí bylo možné opakovaně ověřit funkčnost všech klíčových částí automatizačního procesu.

Byly úspěšně provedeny následující typy testů:

- **Build zlatého obrazu pomocí Packeru** – bylo ověřeno, že nástroj Packer dokáže automatizovaně vytvořit image systému AlmaLinux s požadovanou strukturou disku, předinstalovanými nástroji a podporou cloud-init. Výsledný obraz byl exportován do formátu .box a následně převeden do .ova, připravený pro nasazení do prostředí VMware ESXi.
- **Deploy pomocí Terraform (OpenTofu)** – z předem připraveného .ova obrazu byla vytvořena nová virtuální instance. Bylo potvrzeno, že prostřednictvím YAML konfigurace lze jednoduše přizpůsobit síť, disky i přístupové údaje. V rámci inicializačního skriptu byla použita cloud-init šablona a následný provisioning pomocí Ansible.
- **Instalace a konfigurace Splunk Enterprise** – nástroj byl nasazen zcela automatizovaně. Konfigurace byla vygenerována na základě hodnot ze splunk.yaml.
- **Automatizované vytvoření uživatelů pomocí Terraform provideru** – byla ověřena schopnost připojit se na běžící Splunk instanci přes REST API a založit testovací uživatele s definovanými rolemi na základě CSV souboru.
- **Ověření přihlášení a funkčnosti** – přístup přes webové rozhraní byl úspěšný, nově vytvoření uživatelé byli schopni se přihlásit a pracovat podle svých rolí (admin/power/user).

Celý proces byl opakován několikrát s drobnými úpravami YAML konfigurace, čímž byla potvrzena opakovatelnost a flexibilita navrženého řešení.

Výsledkem je plně funkční automatizovaný build a deployment Splunk infrastruktury, který lze upravovat a rozšiřovat.

5.1 Test běhu Packer build

Výstup zkrácen pro přehlednost. Kompletní log je k dispozici v příloze v souboru

test-reports/packer-example-build.log:

```
[root@docker workspace]# make packer

...
==> vmware-iso.almalinux-9: Retrieving ISO
==> vmware-iso.almalinux-9: Trying
https://ftp.sh.cvut.cz/almalinux/9.6/isos/x86_64/AlmaLinux-9.6-
x86_64-boot.iso
...
==> vmware-iso.almalinux-9: Starting HTTP server on port 8501
==> vmware-iso.almalinux-9: Registering virtual machine on remote
hypervisor...
==> vmware-iso.almalinux-9: Starting virtual machine...
==> vmware-iso.almalinux-9: Connecting to VNC over websocket...
==> vmware-iso.almalinux-9: Waiting 10s for boot...
==> vmware-iso.almalinux-9: Typing the boot command over VNC...
==> vmware-iso.almalinux-9: Waiting for SSH to become available...
==> vmware-iso.almalinux-9: Connected to SSH!
==> vmware-iso.almalinux-9: Provisioning with Ansible...
...
vmware-iso.almalinux-9: PLAY RECAP
*****
*
vmware-iso.almalinux-9: default : ok=35
changed=25 unreachable=0 failed=0 skipped=13 rescued=0
ignored=0
vmware-iso.almalinux-9:
==> vmware-iso.almalinux-9: Gracefully halting virtual machine...
9.6-20250601.x86_64.ovf
==> vmware-iso.almalinux-9: Deleting virtual machine...
==> vmware-iso.almalinux-9: Running post-processor: (type vagrant)
...
Build 'vmware-iso.almalinux-9' finished after 14 minutes 56 seconds.

==> Wait completed after 14 minutes 56 seconds

==> Builds finished. The artifacts of successful builds are:
--> vmware-iso.almalinux-9: 'vmware' provider box: AlmaLinux-9-
Vagrant-vmware-9.6-20250601.x86_64.box
```

5.2 Test běhu Terraform apply – Splunk Universal Forwarder

Výstup zkrácen pro přehlednost. Kompletní log je k dispozici v příloze v souboru

test-reports/tofu-splunkuf-example-apply.log:

```
cd terraform && tofu apply -auto-approve

...
+ ovf_source = "./VMs/alma-base.ova"
...
Plan: 3 to add, 0 to change, 0 to destroy.
...
esxi_guest.alma_vm: Creation complete after 1m58s [id=87]
null_resource.wait_for_ssh: Creating...
null_resource.wait_for_ssh: Provisioning with 'remote-exec'...
...
null_resource.wait_for_ssh (remote-exec): Connected!
null_resource.run_ansible (local-exec): alma :
ok=30 changed=22 unreachable=0 failed=0 skipped=0 rescued=0
ignored=0
null_resource.run_ansible (local-exec): PLAY [Install Splunk Universal
Forwarder] *****
null_resource.run_ansible (local-exec): TASK [Gathering Facts]
*****
null_resource.run_ansible (local-exec): ok: [alma]

null_resource.run_ansible (local-exec): TASK [Enable and start Splunk
UF service] *****
..
null_resource.run_ansible (local-exec): PLAY RECAP
*****
*
null_resource.run_ansible (local-exec): alma :
ok=15 changed=13 unreachable=0 failed=0 skipped=0 rescued=0
ignored=0

null_resource.run_ansible: Creation complete after 1m32s
[id=1262204853455515299]

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.

Outputs:

vm_ip = "192.168.101.30"
vm_name = "alma-51fba"
```

5.2.1 Screenshots z testu Terraform apply

Virtuál vznikl ve VMware s požadovanými vlastnostmi. Splunk UF je nainstalovaný.



alma-51fba

Guest OS	Red Hat Enterprise Linux 9 (64-bit)
Compatibility	ESXi 7.0 U2 virtual machine
VMware Tools	Yes
CPU's	8
Memory	8 GB
Host name	alma-51fba.ptlab.dev

General Information			
Networking			
Host name	alma-51fba.ptlab.dev		
IP addresses	1. 192.168.101.30 2. fe80::20c:29ff:fe65:e17a		
VMware Tools			
Installed	Yes		
Version	open-vm-tools 12.5.0 build 24276846		
Running	Yes		
Storage			
Guest Disks	Disk path	Capacity	Free
	/	8.98 GB	7.92 GB (88%)
	/boot	960 MB	826.1 MB (86%)
	/boot/efi	199.77 MB	192.7 MB (96%)
	/opt/splunk	25.97 GB	25.34 GB (98%)

Obrázek 5 Packer Virtual – Disk

Zdroj: Testovací Build na VMware ESXi

Hardware Configuration	
CPU	8 vCPUs
Memory	8 GB
Hard disk 1	40 GB
Network adapter 1	VLAN 101 - dev (Connected)
Network adapter 2	MGMT (Disconnected)
Network adapter 3	SENSOR (Disconnected)
Video card	8 MB
CD/DVD drive 1	Remote device CD/DVD drive 0
Others	Additional Hardware

Obrázek 6 Packer Virtual – Hardware detail

Zdroj: Testovací Build na VMware ESXi

5.3 Test instalace Splunk Enterprise pomocí Terraform

Výstup zkrácen pro přehlednost. Kompletní log je k dispozici v příloze v souboru

test-reports/tofu-splunk-example-apply.log:

```
[root@docker terraform]# tofu apply
...
# esxi_guest.alma_vm will be created
...
data.template_file.cloudinit: Reading...
esxi_guest.alma_vm: Still creating... [2m20s elapsed]
esxi_guest.alma_vm: Creation complete after 2m22s [id=100]
null_resource.wait_for_ssh: Creating...
...
null_resource.wait_for_ssh (remote-exec): VM is now accessible via SSH
null_resource.wait_for_ssh:      Creation      complete      after      1s
[id=7407410299775214439]
null_resource.run_ansible: Creating...
null_resource.run_ansible      (local-exec):      PLAY      RECAP
*****
*
null_resource.run_ansible (local-exec): alma      :
ok=30  changed=22  unreachable=0  failed=0  skipped=0  rescued=0
ignored=0
null_resource.run_ansible (local-exec):  PLAY  [Install  Splunk
Enterprise] *****
null_resource.run_ansible (local-exec):  TASK   [Download  Splunk
Enterprise] *****
null_resource.run_ansible: Still creating... [1m50s elapsed]
null_resource.run_ansible (local-exec): changed: [alma]

null_resource.run_ansible: Still creating... [5m50s elapsed]
null_resource.run_ansible      (local-exec):      PLAY      RECAP
*****
*
null_resource.run_ansible (local-exec): alma      :
ok=20  changed=15  unreachable=0  failed=0  skipped=0  rescued=0
ignored=0
null_resource.run_ansible:      Creation      complete      after      5m51s
[id=7621000933613196427]

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.

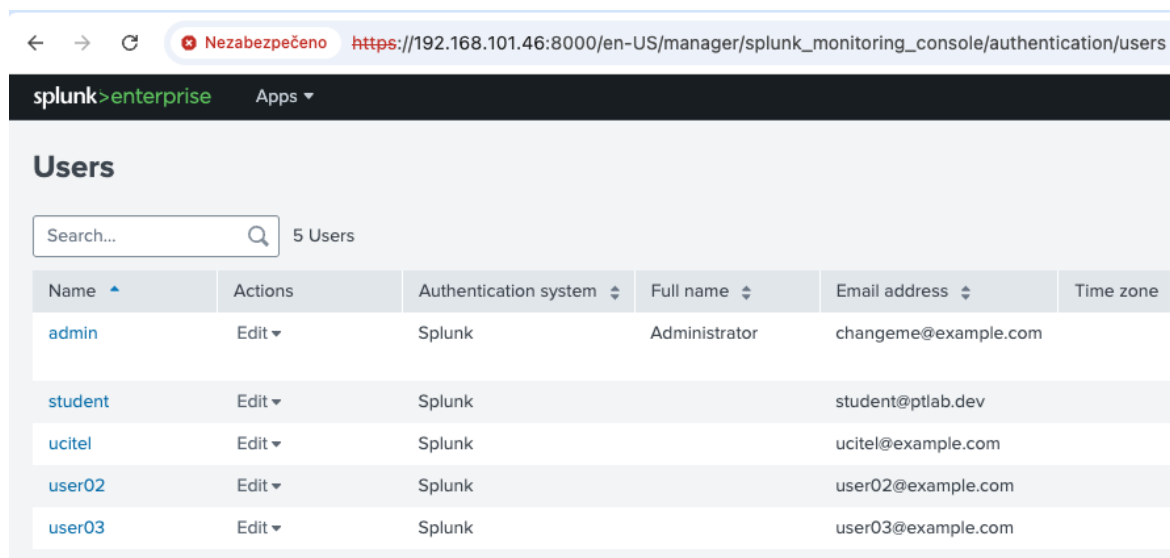
Outputs:
vm_ip = "192.168.101.46"
vm_name = "alma-72c93"
```

5.3.1 Detail testu Splunk Enterprise

Konfigurace: configs/Splunk.yaml

- Lze se přihlásit heslem přes https: ANO
- Splunk verze v pořádku ANO
- Linux kernel transparent huge pages vypnutý: ANO
- HEC Kolektor pro Stream nachystaný: ANO
- Health Check OK: ANO
- Custom indexy surikata a netflow nastaveny (omezení 2 GB): ANO
- Důležité porty běží: ANO (8000, 8089, 8088, 22)
- Lze se přihlásit ssh klíčem: ANO
- Vznikli studenti pomoci users CSV a Terraformu: ANO

```
name,email,password,role
ucitel,ucitel@example.com,password01,admin
user02,user02@example.com,password02,power
user03,user03@example.com,password03,power
```



Obrázek 7 Uživatelé vytvořené přes Terraform/CSV

Zdroj: Build Splunk v testu

ZÁVĚR

Cílem této bakalářské práce bylo navrhnout a realizovat plně automatizované SIEM prostředí, které bude využitelné v rámci výuky kybernetické bezpečnosti. V úvodních částech práce byla provedena analýza dostupných technologií a následně byl vybrán vhodný technologický stack postavený na nástrojích Packer, Terraform (OpenTofu), Ansible a YAML konfiguraci. Volba těchto nástrojů se ukázala jako správná – umožňují vytvořit opakovatelné, flexibilní a transparentní nasazení celého prostředí.

V praktické části se podařilo zcela automatizovaně:

- vytvořit zlatý obraz (Golden Image) systému AlmaLinux 9.6 pomocí Packer,
- přenést výsledný obraz do formátu OVA a nasadit jej do prostředí VMware ESXi,
- pomocí Terraform vytvořit novou instanci a s využitím cloud-init a Ansible provést konfiguraci operačního systému,
- nainstalovat a nakonfigurovat Splunk Enterprise, včetně základního přizpůsobení
- vytvořit Splunk Universal Forwarder a připojit jej k hlavnímu SIEM uzlu,
- pomocí Splunk Terraform provideru definovat testovací uživatele a přiřadit jim role na základě CSV souboru.

V rámci vývoje byl zároveň navržen a vytvořen přenosný vývojový Dev Container, který umožňuje celý proces spustit zcela izolovaně, bez závislosti na konkrétním hostitelském systému. Konfigurace je centralizována do YAML souborů, což výrazně zjednodušuje údržbu a přizpůsobení jednotlivých komponent.

Přestože základní infrastruktura byla úspěšně realizována, na hlubší integraci SIEM funkcionalit (tvorba dashboardů, pravidel, detekcí apod.) již nezbyl dostatek času, a to zejména z důvodu vysoké technické náročnosti prostředí a nedostatečného odhadu časových nároků v rámci plánování.

Výsledné řešení je však plně připravené pro rozšíření – všechny klíčové prvky (build image, infrastruktura, konfigurace, vstupy a uživatelská vrstva) jsou automatizovány a snadno opakovatelné. Práce tak vytváří pevný základ pro další vývoj v rámci diplomové práce, kde se plánuje:

- rozšířit scénáře o simulace bezpečnostních incidentů,
- automatizovat generování testovacích dat (např. pomocí skriptovaných útoků a generátorů provozu),
- implementovat monitorovací pravidla a alerty v rámci Splunk Enterprise Security,
- a doplnit návody pro studenty včetně návrhu cvičení.

Z hlediska praktického přínosu i rozšiřitelnosti se práce naplnila – vytvořené prostředí je využitelné jako základ pro výuku, interní testování i další vývoj bezpečnostních řešení.

SEZNAM POUŽITÉ LITERATURY

- [1] MILLER, David; HARRIS, Shon; HARPER, Allen A.; VANDYKE, Stephen a BLASK, Chris. *Security Information and Event Management (SIEM) Implementation*. New York: McGraw-Hill, 2011. ISBN 978-0-07-170109-9.
- [2] *What is security information and event management (SIEM)?*. Online. In: IBM Think. 2023. Dostupné z: <https://www.ibm.com/think/topics/siem>. [cit. 2025-05-03].
- [3] DAVIES, Andrew; SCHNEIDER, Mitchell; MALIK, Rustam a AHLM, Eric. *Magic Quadrant for Security Information and Event Management*. Online. In: Gartner Reprints. 2024. Dostupné z: <https://www.gartner.com/doc/reprints?id=1-2HJ0VTST&ct=240509&st=s>.
- [4] ABRAHAM, Michelle. *IDC MarketScape: Worldwide SIEM for Enterprise 2024 Vendor Assessment – Google Excerpt*. Online. In: IDC MarketScape. 2024. Dostupné také z: https://idcdocserv.com/US51541324e_Google.
- [5] *The Ultimate Guide to Choosing the Best Linux Distribution for Developers, DevOps Engineers, Pen Testers, and Designers*. Online. In: DEV.to - Olateju Olamide Emmanuel. 2025-02-24. Dostupné z: <https://dev.to/olatejuthedev/the-ultimate-guide-to-choosing-the-best-linux-distribution-for-developers-devops-engineers-pen-419m>. [cit. 2025-05-10].
- [6] *Installation Manual*. Online. In: Splunk Documentation. 2024. Dostupné také z: <https://docs.splunk.com/Documentation/Splunk/9.4.2/Installation>.
- [7] *Announcing the General Availability of Splunk® Enterprise Security 8.0*. Online. In: Splunk Blog. 2024-10-31. Dostupné z: https://www.splunk.com/en_us/blog/security/announcing-the-general-availability-of-splunk-enterprise-security-8-0.html. [cit. 2025-05-03].
- [8] *Splunk Enterprise Overview*. Online. In: Splunk Documentation. 2025-02-07. Dostupné z: <https://docs.splunk.com/Documentation/Splunk/9.4.2/Overview>. [cit. 2025-05-03].
- [9] *Splunk® Enterprise Security*. Online. In: Splunk Documentation. Dostupné také z: <https://docs.splunk.com/Documentation/ES/8.0.40>.

- [10] *Install and Upgrade Splunk Enterprise Security*. Online. In: Splunk Documentation. Dostupné z: <https://docs.splunk.com/Documentation/ES/8.0.40/Install>. [cit. 2025-05-03].
- [11] *Documentation Home Page*. Online. In: Splunk Documentation. Dostupné z: <https://docs.splunk.com/Documentation>. [cit. 2025-05-03].
- [12] *Docker Splunk - Release 9.4.1*. Online. In: Splunk GitHub. 2025-02-26. Dostupné z: <https://github.com/splunk/docker-splunk/releases/tag/9.4.1>. [cit. 2025-05-03].
- [13] *What is Microsoft Sentinel?* Online. In: Microsoft Learn. 2024-05-21. Dostupné z: <https://learn.microsoft.com/en-us/azure/sentinel/overview>. [cit. 2025-05-03].
- [14] *Prerequisites for installing QRadar on your hardware*. Online. In: IBM Documentation. 2025-04-24. Dostupné z: <https://www.ibm.com/docs/en/qsip/7.5?topic=installations-prerequisites-installing-qradar-your-hardware>. [cit. 2025-05-10].
- [15] *Installing and Upgrading*. Online. In: Securonix On-Prem Documentation. 2025-04-28. Dostupné z: <https://documentation.securonix.com/bundle/securonix-on-prem-user-guide/page/content/installing-and-upgrading.htm>. [cit. 2025-05-11].
- [16] *Exabeam and LogRhythm Merge, Becoming a Pure-Play SecOps Leader*. Online. In: Exabeam Resources. 2024-11-18. Dostupné z: <https://www.exabeam.com/resources/briefs/exabeam-and-logrhythm-merge-becoming-a-pure-play-secops-leader>. [cit. 2025-05-11].
- [17] *NetMon Reference Architecture*. Online. In: LogRhythm NetMon Docs. 2025-04-01. Dostupné z: <https://docs.logrhythm.com/netmon/docs/netmon-reference-architecture>. [cit. 2025-05-11].
- [18] *The intelligence-driven and AI-powered security operations platform*. Online. In: Google Cloud. 2025. Dostupné z: <https://cloud.google.com/security/products/security-operations>. [cit. 2025-05-11].
- [19] *Devo Platform Packaging*. Online. In: DEVO. 2025. Dostupné z: <https://www.devo.com/packaging/>. [cit. 2025-05-11].
- [20] *Elastic Support*. Online. In: Elastic. 2025-04-08. Dostupné z: <https://www.elastic.co/support>. [cit. 2025-05-11].

- [21] *SOAR for the modern SOC*. Online. In: Elastic Security. 2025. Dostupné z: <https://www.elastic.co/security/soar>. [cit. 2025-05-11].
- [22] *Deploy and manage*. Online. In: Elastic Docs. 2025. Dostupné z: <https://www.elastic.co/docs/deploy-manage/>. [cit. 2025-05-11].
- [23] *Elastic Security Docs*. Online. In: Elastic Solutions and use cases. 2025. Dostupné z: <https://www.elastic.co/docs/solutions/security>. [cit. 2025-05-11].
- [24] *ArcSight ESM 7.8 Technical Requirements*. Online. In: *Open Text Corporation Documentation*. 2024-08. Dostupné z: https://www.microfocus.com/documentation/arcsight/arcsight-esm-7.8/pdfdoc/ESM_Support_Matrix/ESM_Support_Matrix.pdf. [cit. 2025-05-11].
- [25] *Review the Requirements for a New LogRhythm Deployment*. Online. In: *LogRhythm Installations and Upgrades*. 2025. Dostupné z: <https://docs.logrhythm.com/deploy/docs/review-the-requirements-for-a-new-logrhythm-deploy>. [cit. 2025-05-11].
- [26] *InsightIDR Overview*. Online. In: *Rapid7 Documentation*. 2025. Dostupné z: <https://docs.rapid7.com/insightidr/>. [cit. 2025-05-11].
- [27] BRIKMAN, Yevgeniy. *Terraform: up & running: writing infrastructure as code*. Third edition. Beijing: O'Reilly, 2022. ISBN 978-1-098-11674-3.
- [28] WOJSŁAW, Damian a ADAMOWICZ, Grzegorz. *The Linux DevOps handbook: customize and scale your Linux distributions to accelerate your DevOps workflow*. Birmingham: Packt, [2023]. ISBN 978-1-80324-566-9.
- [29] *HCL Configuration Language*. Online. In: *Packer Developer Docs*. 2025. Dostupné také z: https://developer.hashicorp.com/packer/docs/v1.12.x/templates/hcl_templates. [cit. 2025-05-31].
- [30] *Splexicon the Splunk glossary*. Online. In: *Splunk Documentation*. 2025. Dostupné z: <https://docs.splunk.com/Splexicon>. [cit. 2025-05-31].
- [31] *INI - Initiation File Format*. Online. In: *FILEFORMAT Documentation*. 2025. Dostupné také z: <https://docs.fileformat.com/system/ini/>. [cit. 2025-05-31].

- [32] *Empower Learning with the Splunk Academic Alliance*. Online. In: Splunk Resources. 2025. Dostupné z: https://www.splunk.com/en_us/resources/splunk-academic-alliance.html. [cit. 2025-05-31].
- [33] *Dev Containers tutorial*. Online. In: Visual Studio Code Docs. 2025. Dostupné z: <https://code.visualstudio.com/docs/devcontainers/tutorial>. [cit. 2025-05-31].
- [34] *Flowcharts - Basic Syntax*. Online. In: Mermaid Diagram Syntax. 2025, 2025-05-18. Dostupné z: <https://mermaid.js.org/syntax/flowchart.html>. [cit. 2025-05-31].
- [35] *Repository – terraform-provider-esxi*. Online. In: josenk GitHub. 2024, 2024-10-11. Dostupné z: <https://github.com/josenk/terraform-provider-esxi>. [cit. 2025-05-31].
- [36] *Repository – cloud-images*. Online. In: AlmaLinux GitHub. 2025, 2025-05-29. Dostupné z: <https://github.com/AlmaLinux/cloud-images>. [cit. 2025-05-31].
- [37] *DetectionLab - Introduction*. Online. In: INTERNET ARCHIVE - Wayback Machine. 2024-03-03. Dostupné z: <https://web.archive.org/web/20240303142926/https://detectionlab.network/introduction/>. [cit. 2025-05-18].

SEZNAM OBRÁZKŮ

Obrázek 1 IDC MarketScape Worldwide SIEM for Enterprise	26
Obrázek 2 Packer	31
Obrázek 3 Architektura SIEM prostředí rozdělená do tří hlavních vrstev.	43
Obrázek 4 Suricata - Parsing JSON	60
Obrázek 5 Packer Virtual – Disk	64
Obrázek 6 Packer Virtual – Hardware detail	64
Obrázek 7 Uživatelé vytvořené přes Terraform/CSV	66

SEZNAM TABULEK

Tabulka 1 Srovnání SIEM.....	25
Tabulka 2 Technologie pro automatizace	33
Tabulka 3 Hardwarové požadavky	38
Tabulka 4 Softwarové požadavky	39
Tabulka 5 Konfigurace sítě pro jednotlivé služby	40

SEZNAM PŘÍLOH

PŘÍLOHA P I: MERMAID DIAGRAM ARCHITEKTURY

PŘÍLOHA ZIP:

- Vývojové a instalační soubory
- Obsahuje kompletní balík pro reprodukci řešení:
- vývojové prostředí (Dev Container, Dockerfile, VS Code konfigurace),
- instalační nástroj ovftool,
- skripty a šablony nástrojů Packer, Terraform, Ansible,
- ukázkové konfigurační soubory a Makefile pro řízení sestavení,
- vývojový repozitář ve struktuře použité v práci,
- předpřipravený Windows 10 image s nainstalovaným Splunk Universal Forwarderem a zkušební (evaluation) licencí Splunk Enterprise (90 dní).

Příloha bude předána jako jeden komprimovaný archiv (prilohy.zip).

PŘÍLOHA P I: MERMAID DIAGRAM ARCHITEKTURY

```
1 graph TD
2     %% Vrstvy
3     subgraph "(1) Vývoj a administrace"
4         subgraph "Učitel"
5             Ucitel["Pracovní stanice (notebook)"]
6         end
7         subgraph "Docker "
8             DevContainer["Dev Container"]
9         end
10    end
11    subgraph "(3) Uživatelé a síť"
12        subgraph "Učebna"
13            UcebnaPC["Win pC"]
14        end
15    end
16    subgraph "(2) Infrastruktura"
17        subgraph "VMware ESXi"
18            subgraph "SIEM Virtual Server"
19                SIEM["Splunk Enterprise"]
20                Suricata["Suricata IDS"]
21                Splunk_Stream_Forwarder["Splunk Stream Forwarder"]
22            end
23            subgraph "Serving"
24                WinServer["Windows Server"]
25                subgraph "Linux Server"
26                    Splunk_Stream_Forwarder2["Splunk Stream Forwarder"]
27                    AlmaLinux["Alma Linux"]
28                end
29            end
30        end
31        subgraph "Networking"
32            Flowmon["Flowmon"]
33            Switch["Switch"]
34        end
35    end
36
37    Ucitel --> DevContainer
38    DevContainer --> AlmaLinux
39    DevContainer --> WinServer
40    DevContainer --> SIEM
41    WinServer --> SIEM
42    AlmaLinux --> SIEM
43    Switch --> Suricata
44    Flowmon --> Splunk_Stream_Forwarder
45    Splunk_Stream_Forwarder --> SIEM
46    Splunk_Stream_Forwarder2 --> SIEM
47    Suricata --> SIEM
48    UcebnaPC --> SIEM
49    UcebnaPC --> WinServer
50    UcebnaPC --> AlmaLinux
```