PROJEKT

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

V16 Funkčný PoC

KPB5 – Realizácia riešenia

typ – dokumentácia

mesiac odovzdania – M22

Financované Európskou úniou
NextGenerationEU

PLÁN OBNOVY

ÚRAD VLÁDY
SLOVENSKEJ REPUBLIKY

Obsah

1 Úvod 4
1.1 Účel dokumentu 4
1.2 Východiská a vstupy do PoC 4
1.3 Rozsah PoC 4
1.4 Prevádzkové prostredie PoC 4
1.5 Obmedzenia PoC 5
2 Implementované komponenty v PoC INNOSOC 6
2.1 SIEM - ELK Stack 6
2.1.1 Model nasadenia ELK Stack 6
2.1.2 Nasadenie Elasticsearch 6
2.1.3 Nasadenie Logstash 8
2.2 Shuffle 18
2.2.1 Model nasadenia Shuffle 18
2.2.2 Základné integračné väzby komponentu Shuffle v PoC 19
2.3 Wazuh 20
2.3.1 Model nasadenia komponentu Wazuh 20
2.3.2 Monitorované prostredia nástrojom Wazuh 20
2.3.3 Nasadené a aktivované moduly systému Wazuh 23
2.3.4 Základné integračné väzby - posielanie notifikácií 24
2.4 Greenbone Community edition 31
2.5 OpenCTI 35
2.5.1 Nasadenie Shuffle v prostredí OpenStack 35
2.5.2 Základné integračné väzby pre OpeCTI komponent 36
2.6 MISP ..... **Chyba! Záložka nie je definovaná.**
2.7 GLPI 37
2.8 NetBox 42
2.8.1 Úloha NetBox v PoC 42
2.8.2 Nasadenie a základná konfigurácia NetBoxu 42
2.8.3 Využitie NetBoxu v rámci PoC 44
2.8.4 Aktuálny stav a obmedzenia NetBoxu v PoC 48
2.9 DFIR IRIS 48
2.9.1 Model nasadenia DFIR IRIS v prostredí OpenStack 48
2.9.2 Základné integračné väzby komponentu DFIR IRIS v INNOSOC 49
2.10 Kibana (ELK) 50
2.10.1 Vytvorené dashboardy 50

2.10.2 Technické parametre a overenie funkčnosti 53
2.11 Wiki.js 54
3 Nasadenie a základná konfigurácia PoC ..... Chyba! Záložka nie je definovaná.
3.1 Model nasadenia ..... **Chyba! Záložka nie je definovaná.**
3.2 Použité virtuálne stroje a služby ..... **Chyba! Záložka nie je definovaná.**
3.3 Základné integračné väzby ..... **Chyba! Záložka nie je definovaná.**
4 Záver 61
4.1 XX ..... **Chyba! Záložka nie je definovaná.**

1 ÚVOD

1.1 Účel dokumentu

Účelom tohto dokumentu je zdokumentovať funkčný PoC v rámci pracovného balíka KPB5 – Realizácia riešenia projektu INNOSOC. Dokument nadväzuje na predchádzajúce architektonické a realizačné výstupy projektu a sumarizuje stav, v ktorom je navrhované riešenie implementované ako funkčný experimentálny koncept.

Dokument zachytáva funkčný stav realizovaného riešenia ako integrovaného celku, teda ako multitenantnej cloudovej služby typu SOC, ktorá je nasadená v experimentálnom režime a pripravená na formálne overenie. Jeho hlavným cieľom je preukázať, že navrhované riešenie bolo implementované do funkčnej a prevádzkyschopnej podoby v rozsahu PoC a predstavuje technickú a prevádzkovú dokumentáciu experimentálne vytvoreného riešenia..

1.2 Východiská a vstupy do PoC

PoC vychádza z výsledkov predchádzajúcich pracovných balíkov a z nadväzujúcich výstupov projektu. Základný rámec realizácie bol vytvorený v KPB4 – Architektúra riešenia, ktorého cieľom bolo na základe predchádzajúcich analytických a návrhových výstupov vytvoriť ucelený technologický koncept. V rámci KPB4 boli definované funkčné a nefunkčné požiadavky, biznis architektúra, technologická architektúra a finálny DNR – Detailný návrh riešenia, ktoré predstavujú priamy vstup do implementačnej etapy KPB5.

Za vecné vstupy do realizácie PoC je potrebné považovať aj výsledky predchádzajúcich návrhových a analytických činností projektu, najmä návrh virtualizačného riešenia, návrh služby SOCaaS a návrh integračnej architektúry jednotlivých komponentov. Projektový zámer medzi hlavné úlohy realizácie radí konfiguráciu infraštruktúry, inštaláciu softvérových aplikácií, ich integráciu a testovanie navrhovaného riešenia. PoC vzniká ako výsledok postupnej nadväznosti na predchádzajúce architektonické, technologické a infraštruktúrne výstupy projektu.

1.3 Rozsah PoC

Tu opísaný PoC zachytáva implementované časti navrhovaného riešenia v rozsahu potrebnom na overenie základnej funkčnosti služby. Predmetom tohto výstupu je najmä opis nasadených komponentov, ich základných väzieb a prevádzkového usporiadania v prostredí projektu. Dokument sa sústreďuje na to, čo bolo v rámci KPB5 reálne nasadené, nakonfigurované a overené. PoC nepredstavuje finálne produkčné riešenie ani kompletnú komerčnú službu. Ide o experimentálny koncept, ktorého účelom je preukázať technickú realizovateľnosť navrhovanej architektúry a základnú prevádzkyschopnosť riešenia ako celku. Rozsah PoC je preto vedome obmedzený na funkcie, ktoré sa podarilo implementovať a overiť v laboratórnych podmienkach.

1.4 Prevádzkové prostredie PoC

PoC je prevádzkovaný nad privátnym cloudovým prostredím OpenStack, ktoré tvorí základnú infraštruktúrnu vrstvu riešenia. Jednotlivé komponenty sú nasadené vo virtuálnych strojoch, pričom podľa charakteru konkrétneho nástroja je využitá aj kontajnerizácia prostredníctvom Docker a Docker Compose. Tento model umožňuje oddeliť jednotlivé funkčné bloky riešenia, zjednodušiť ich správu a zároveň zachovať flexibilitu pri experimentálnom

nasadzovaní. Prevádzkové prostredie PoC má charakter testovacieho a overovacieho prostredia. Jeho účelom nie je poskytovanie plnohodnotnej produkčnej služby, ale vytvorenie funkčnej platformy na integráciu, konfiguráciu a overenie navrhovaných komponentov a ich vzájomnej spolupráce. Prostredie je preto navrhnuté tak, aby umožňovalo opakované nasadenie, úpravy konfigurácie a realizáciu testovacích scenárov.

1.5 Obmedzenia PoC

PoC je poskytuje experimentálny dôkaz koncepcie. Nie všetky plánované komponenty a integračné väzby boli v čase spracovania tohto výstupu dotiahnuté do finálnej úrovne funkčnosti. V niektorých prípadoch boli komponenty úspešne nasadené samostatne, avšak ich integrácia zostala len čiastočne realizovaná. Dokument tak zachytáva reálny stav implementácie, nie finálny stav architektúry. Obmedzenia sa týkajú najmä miery dokončenia jednotlivých integrácií, rozsahu overených scenárov a celkovej úrovne prevádzkovej zrelosti riešenia. PoC však poskytuje dostatočný základ na technické overenie konceptu a na následné vyhodnotenie dosiahnutých výsledkov v ďalšom výstupe projektu.

.

BRAIN:IT

FRI

2 IMPLEMENTOVANÉ KOMPONENTY V POC INNOSOC

2.1 SIEM - ELK Stack

2.1.1 Model nasadenia ELK Stack

ELK je v rámci PoC riešenia INNOSOC využitý ako SIEM, ktorý sa skladá z Elasticsearch indexovej databázy, Logstash parsovacieho nástroja, ktorý predpripravuje formát dát, pred uložením do databázy, Kibany, ktorá slúži ako WEB GUI pre celý stack. Kibana zobrazuje nazbierané dáta, poskytuje správu používateľov a rolí, tvorbu bezpečnostných pravidiel, riešenie alertov, správu indexov, správu uzlov (nodes) a iné.

Celý ELK stack je nasadený v LXC linuxových kontajneroch. Pre každú službu je vytvorený samostatný kontajner. Operačný systém nasadený v kontajneroch je Ubuntu 24.04.2.

Všetky LXC kontajnery sú nasadené na jednom fyzickom serveri. Server má procesor Intel(R) Xeon(R) CPU E5-2690, 2 fyzické procesory, 8 jadier na procesor s frekvenciou 2.90GHz. Server disponuje s 251GB RAM pamäte. Úložný priestor je tvorený raidom HDD diskov o veľkosti 2.7T a SSD diskom 1.8T WD Red.

2.1.2 Nasadenie Elasticsearch

Elasticsearch kontajner má IP adresu 192.168.1.178. Beží na ňom Elasticsearch cluster, ktorý je tvorený 7 elasticsearch uzlami. Uzly majú role master, hot, warm a cold. Master node je hlavný node, ostatné data nodes sa naň pripájajú a začleňujú sa do clustera. Hot nodes sú uzly, kde sa uchovávajú najaktuálnejšie dáta. Dáta sa ukladajú na SSD disk, aby sa stíhali zapisovať v čo najkratšom časovom intervale. Tieto dáta sa tam uchovávajú po dobu 5 dní. Po tejto dobe sa presúvajú na warm nodes. Warm nodes sú 2 a používajú HDD disky. Dáta sú uložené vo warm nodes po dobu od 5 do 14 dní. Následne sa dáta presúvajú do cold uzla a po 21 dňoch sa mažú.

Node	Disky	Retenčná doba	Pridelená RAM
master1	HDD	---	16GB
hot1	SSD	<5 dní	31GB
hot2	SSD	<5 dní	31GB
hot3	SSD	<5 dní	31GB
warm1	HDD	<14 dní	31GB
warm2	HDD	<14 dní	31GB
cold1	HDD	<21 dní	16GB

V projekte je kladený dôraz na zabezpečenie. Elasticsearch predvolene používa, od verzie 7, x-pack modul, ktorý zabezpečuje viacero bezpečnostných funkcií, vrátane autentifikácie, autorizácie, šifrovania komunikácie a správy používateľov.

Komunikácie je medzi uzlami šifrovaná pomocou TLS certifikátov. Každý node má svoj TLS certifikát. Komunikácie medzi Logstash a Elastich search aj Kibana a Elasticsearch sú taktiež šifrované. Každý Elasticsearch uzol má vlastný konfiguračný súbor.

Použitá certifikačná autorita (CA), ktorá vystavuje jednotlivé certifikáty, je vlastná. Elastic poskytuje možnosť vygenerovať CA pomocou nástroja *elasticsearch-certutil*, ktorý je súčasťou distribúcie *Elasticsearch*. Tento nástroj umožňuje vytvárať certifikát certifikačnej autority, ale aj vystavovať ďalšie certifikáty podpísané touto certifikačnou autoritou.

Elasticsearch okrem toho využíva tzv. keystore. Keystore slúži ako úložisko pre citlivé údaje, ako sú prihlasovacie údaje, tokeny alebo heslá systémových používateľov. V našom prípade využívame keystore na uloženie prihlasovacích údajov preddefinovaných systémových účtov. Tento mechanizmus zvyšuje bezpečnosť systému tým, že citlivé údaje nie sú priamo uložené v konfiguračných súboroch.

Okrem zabezpečenia citlivých údajov pomocou keystore je potrebné zabezpečiť aj sieťovú komunikáciu samotného *elk-elasticsearch* kontajneru. Na tento účel sme využili nástroj iptables. Pomocou neho sme definovali firewall pravidlá. Prístup pre službu Secure Shell (SSH) je povolený z univerzitného rozsahu a prístup pre *Elasticsearch* aplikáciu je povolený len zo strojov *Kibana*, *Logstash* a *Arkime*, definované na základe IP adries.

V rámci monitorovania dostupnosti bolo okrem, už spomínaného, monitorovania ELK stack pomocou metricbeat zavedené monitorovanie pomocou nástroja libreNMS a pomocou nástroja zabbix.

2.1.2.1 Ukážka konfiguračného súboru hot uzla

```
cluster.name: elastic-cluster
node.name: hot1
node.roles: [data_hot]
path.data: /mnt/hot-nodes/elasticsearch-hot1
path.logs: /var/log/elasticsearch-hot1
network.host: 192.168.1.178
http.port: 9201
transport.port: 9301
discovery.seed_hosts:  ["192.168.1.178:9300",  "192.168.1.178:9302",
"192.168.1.178:9303",  "192.168.1.178:9304",  "192.168.1.178:9305",
"192.168.1.178:9306"]
cluster.initial_master_nodes: ["master1"]
xpack.security.enabled: true
xpack.security.enrollment.enabled: true
xpack.security.http.ssl:
  enabled: true
  keystore.path: certs/elasticsearch/hot1/http.p12
xpack.security.transport.ssl:
  enabled: true
  verification_mode: certificate
  client_authentication: required
  keystore.path: certs/elastic-certificates.p12
  truststore.path: certs/elastic-certificates.p12
http.host: 0.0.0.0
transport.host: 0.0.0.0
xpack.security.audit.enabled: true
xpack.security.audit.logfile.events.include:          access_denied,
anonymous_access_denied,  authentication_failed,  connection_denied,
tampered_request, run_as_denied, security_config_change
xpack.security.audit.logfile.emit_node_host_address: true
```

```
xpack.security.audit.logfile.emit_node_name: true
xpack.security.audit.logfile.emit_node_host_name: true
```

2.1.3 Nasadenie Logstash

Logstash spracováva dáta prijaté od auditbeat a filebeat agentov, ktorí sú nasadení na ostatných serveroch na FRI UNIZA. Komunikácia medzi agentami a Logstash je taktiež šifrovaná. Logstash pipeline je rozdelená do troch konfiguračných súboroch. Pre vstup, filtrovanie a výstup zozbieraných dát.

Pre vstup, filtrovanie a výstup zozbieraných dát:

- input.conf – definuje zdroje prijatých dát,
- filters.conf – spracováva prijaté dáta, aplikuje na dáta filtre, rieši normalizáciu dát
- output.conf – spravuje odoslanie dát.

Obdobne ako pri nástroji Elasticsearch aj pre nástroj Logstash sme vykonali čistú inštaláciu. Na stroji ucs-lxc-1 s IP adresou 192.168.2.11 bol pre potreby nasadenia Logstash nástroja vytvorený LXC kontajner s operačným systémom Ubuntu 24.04 LTS. Kontajner je pomenovaný elk-logstash s pridelenou IP adresu 192.168.1.177.

Logstash je nasadený vo verzii 8.17.2. Pri jeho konfigurácii je definovaná veľkosť operačnej pamäte, obdobne ako pri Elasticsearch uzloch, pričom je alokovaných 16 GB. Táto hodnota poskytuje dostatočnú rezervu aj pre budúce rozšírenie infraštruktúry a pripojenie ďalších monitorovaných zariadení.

Komunikácia Logstash a Elasticsearch je zabezpečená certifikátom na strane Elasticsearch. Overenie Logstash voči Elasticsearch prebieha pomocou mena a hesla. Keďže sme sa snažili dodržať princíp najnižších potrebných oprávnení tzv. „least privilege“, bolo potrebné vytvoriť používateľa ktorému bola pridelená rola s oprávneniami. Vytvorená rola nesie názov logstash_writer a vytvorený používateľ logstash_internal. Danej roli sme pridelili nasledujúce oprávnenia pre cluster:

- monitor - sledovanie stavu klastra a uzlov
- manage_index_template - správa šablón indexov
- manage_ilm - správa životného cyklu indexov
- manage_pipeline - správa ingest pipeline

A oprávnenia pre jednotlivé indexy:

- write - zapisovanie dát do indexu
- create - vytváranie nových dokumentov
- create_index - vytváranie nových indexov
- manage - plná správa indexu
- read - čítanie dát z indexu
- index - pridávanie alebo aktualizácia dokumentov

Indexy pre ktoré sa majú privilégia prideliť sme definovali pomocou tzv. index pattern. Index pattern je vzor alebo pravidlo, ktoré definuje konkrétny index. Využívajú sa v nich tzv. wildcards, v podobe znaku „ * “, spolu v kombinácií textom, ktorý tvorí časť názvu indexu. Pre našu rolu sme definovali nasledujúce index pattern:

1. gvm-greenbone*
2. elasticsearch-elk-elasticsearch*
3. syslog-fortigate*
4. waf-filebeat*

5. auditbeat*
6. suricata*
7. threatintel-filebeat*

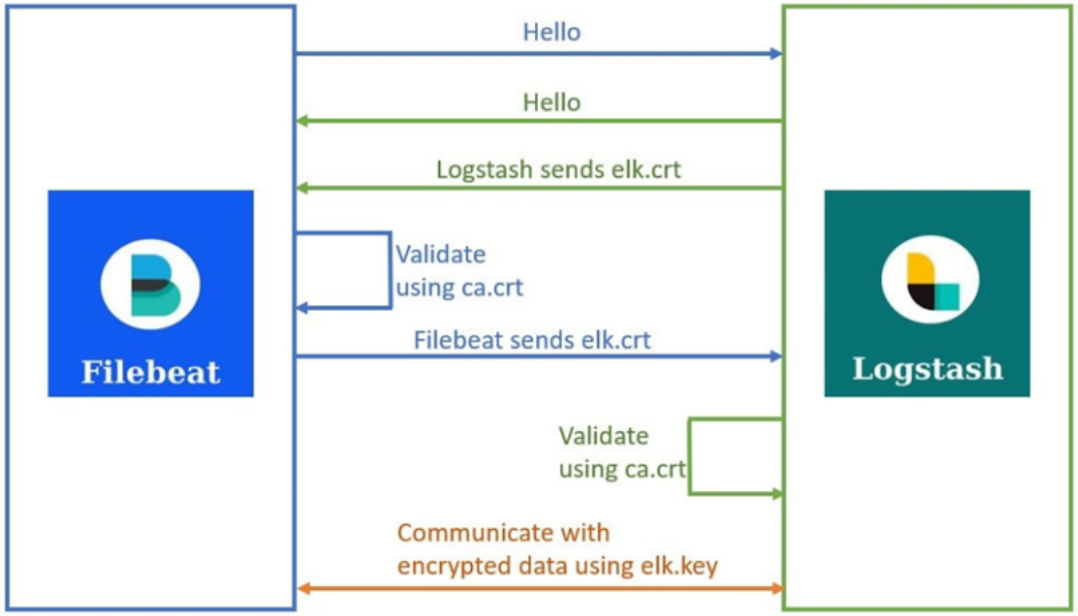
Logstash využíva keystore na uloženie hesla používateľa pre Logstash API a na uloženie hesla pre používateľa logstash_internal. Sieťová komunikácia je zabezpečená pomocou nástroja iptables, ktorým sme definovali firewall pravidlá. Prístup pre službu SSH a pre príjem dát pomocou Logstash je povolený zo špecifického IPv4 rozsahu.

Logstash je v rámci monitorovania dostupnosti monitorovaný pomocou metricbeat a taktiež pomocou nástrojov zabbix a libreNMS.

2.1.3.1 Ukážka input.conf konfiguračného súboru pre nasadený Logstash

V konfiguračnom súbore je nastavený príjem dát pomocou modulu beat, ktorý dáta prijíma na porte 5044. Ďalej je nastavené Secure Sockets Layer (SSL)/ Transport Layer Security (TLS) šifrovanie a definované cesty k certifikátom. Certifikáty boli podpísané vlastnou certifikačnou autoritou. Logstash vyžaduje aby sa klienti preukázali platným certifikátom, inak spojenie nebude nadviazané, a taktiež poskytuje vlastný TLS certifikát ktorým si klienti overia identitu Logstash. Ide o obojstranné TLS overovanie.

```
input {
        beats {
                port => 5044
                ssl_enabled => true
                ssl_certificate                                         =>
            "/etc/logstash/certs/logstash/logstash.crt"
                ssl_key => "/etc/logstash/certs/logstash/logstash.key"
                ssl_certificate_authorities                             =>
            ["/etc/logstash/certs/ca.crt"]
                ssl_client_authentication => "required"
        }
}
```



Znázornenie obojstranného TLS

2.1.3.2 Ukážka filters.conf konfiguračného súboru pre nasadený Logstash

```
filter { if ("suricata-moloch" in [tags]) or ("suricata-suricata2"
in [tags]) { json { source => "[event][original]" target =>
"suricata_parsed" } date { match => ["[suricata_parsed][timestamp]",
"yyyy-MM-dd'T'HH:mm:ss.SSSSSSZ"] target => "@timestamp" } mutate {
rename => { "[suricata_parsed][src_ip]" => "source.ip"
"[suricata_parsed][src_port]" => "source.port"
"[suricata_parsed][dest_ip]" => "destination.ip"
"[suricata_parsed][dest_port]" => "destination.port"
"[suricata_parsed][proto]" => "network.protocol"
"[suricata_parsed][event_type]" => "event.type"
"[suricata_parsed][flow_id]" => "network.flow_id"
"[suricata_parsed][app_proto]" => "network.application_protocol"
"[suricata_parsed][in_iface]" => "network.interface"
"[suricata_parsed][alert][signature]" => "rule.name"
"[suricata_parsed][alert][signature_id]" => "rule.id"
"[suricata_parsed][alert][category]" => "rule.category"
"[suricata_parsed][alert][severity]" => "rule.severity"
"[suricata_parsed][alert][action]" => "rule.action"
"[suricata_parsed][alert][rev]" => "rule.revision"
"[suricata_parsed][alert][gid]" => "rule.gid"
"[suricata_parsed][flow][pkts_toserver]" =>
"network.packets.to_server" "[suricata_parsed][flow][pkts_toclient]"
=> "network.packets.to_client"
"[suricata_parsed][flow][bytes_toserver]" =>
"network.bytes.to_server" "[suricata_parsed][flow][bytes_toclient]"
=> "network.bytes.to_client" "[suricata_parsed][flow][start]" =>
"network.flow_start"
"[suricata_parsed][alert][metadata][signature_severity]" =>
```

```
"rule.signature_severity"
"[suricata_parsed][alert][metadata][created_at]" =>
"rule.created_at" "[suricata_parsed][alert][metadata][updated_at]"
=> "rule.updated_at" "[suricata_parsed][alert][metadata][cve]" =>
"rule.cve_list" "[suricata_parsed][alert][metadata][deployment]" =>
"rule.deployment"
"[suricata_parsed][alert][metadata][attack_target]" =>
"rule.attack_target"
"[suricata_parsed][alert][metadata][affected_product]" =>
"rule.affected_product" "[suricata_parsed][alert][metadata][tag]" =>
"rule.tag" } convert => { "network.flow_id" => "integer"
"source.port" => "integer" "destination.port" => "integer"
"network.packets.to_server" => "integer" "network.packets.to_client"
=> "integer" "network.bytes.to_server" => "integer"
"network.bytes.to_client" => "integer" "rule.id" => "integer"
"rule.severity" => "integer" "rule.revision" => "integer" "rule.gid"
=> "integer" } } geoip { source => "source.ip" target => "source"
database => "/etc/logstash/geoip/GeoLite2-City.mmdb" } geoip {
source => "destination.ip" target => "destination" database =>
"/etc/logstash/geoip/GeoLite2-City.mmdb" } mutate { remove_field =>
[ "source.ip", "destination.ip"] rename => {
"[suricata_parsed][src_ip]" => "source.ip"
"[suricata_parsed][dest_ip]" => "destination.ip" } } mutate {
remove_field => [ "message", "suricata_parsed"] } }

else if ("auditbeat-studenti" in [tags]) or ("auditbeat-admin" in
[tags]) or ("auditbeat-testing" in [tags]) or ("auditbeat-elk" in
[tags]){ mutate { add_field => { "[@metadata][index]" => "auditbeat-
logs" } add_field => { "event.ingested" => "%{+YYYY-MM-
dd'T'HH:mm:ss.SSSZ}" } rename => {
"[process][parent][process][name]" => "process.parent.name"
"[process][parent][process][executable]" =>
"process.parent.executable" "[process][parent][process][args]" =>
"process.parent.args" "[process][parent][process][pid]" =>
"process.parent.pid" "[process][parent][process][entity_id]" =>
"process.parent.entity_id" "[process][parent][process][title]" =>
"process.parent.title" "[process][parent][process][start_time]" =>
"process.parent.start_time" "[process][parent][process][owner]" =>
"process.parent.owner" "[process][parent][process][group]" =>
"process.parent.group" "[process][parent][process][thread]" =>
"process.parent.thread" } } mutate { replace => {
"process.parent.pid" => "%{[process][parent][pid][0]}" } }

mutate {
  remove_field => ["process.parent.pid"]
}
mutate {
  convert => { "process.parent.pid" => "integer" }
}
if [process][args] and ( [process][args] != "" ) {

  mutate {
```

```
add_field => { "[process][command_line]" => "%{[process][args]}" }
  }
  mutate {
    gsub => [
      "[process][command_line]", '\"', '',
      "[process][command_line]", '\[', '',
      "[process][command_line]", '\]', '',
      "[process][command_line]", ',', ' '
    ]
  }
}


} else if "syslog-fortigate" in [log_type] { grok { match => {
"message" => "%{SYSLOGTIMESTAMP:syslog_timestamp}
%{IPORHOST:syslog_host} %{GREEDYDATA:event.original}" } overwrite =>
["event.original"] } kv { source => "event.original" field_split =>
" " value_split => "=" trim_key => """ trim_value => """ } mutate {
rename => { "srcip" => "source.ip" "dstip" => "destination.ip" }
convert => { "srcport" => "integer" "dstport" => "integer"
"sentbyte" => "integer" "rcvdbyte" => "integer" "sentpkt" =>
"integer" "rcvdpkt" => "integer" "duration" => "integer" } add_field
=> { "datetime" => "%{date} %{time}" } } date { match =>
["datetime", "yyyy-MM-dd HH:mm:ss"] target => "@timestamp" timezone
=> "+0100" } mutate { remove_field => ["datetime", "message",
"date"] } if [agent] { mutate { add_field => { "agent_name" =>
"%{[agent][name]}" "agent_id" => "%{[agent][id]}" "agent_type" =>
"%{[agent][type]}" "agent_version" => "%{[agent][version]}"
"agent_ephemeral_id" => "%{[agent][ephemeral_id]}" } } mutate {
remove_field => ["agent"] }

 } } else if "CTF-win" in [tags]{ mutate { add_field => {
"[@metadata][index]" => "ctf-win-logs" } add_field => {
"event.ingested" => "%{+YYYY-MM-dd'T'HH:mm:ss.SSSZ}" } } if
[winlog][provider_name] == "Microsoft-Windows-Sysmon" { mutate {
rename => { "[winlog][event_id]" => "event.sysmon_id"
"[winlog][event_data][Image]" => "process.name"
"[winlog][event_data][CommandLine]" => "process.command_line"
"[winlog][event_data][ParentImage]" => "process.parent.name"
"[winlog][event_data][ParentCommandLine]" =>
"process.parent.command_line" "[winlog][event_data][User]" =>
"user.name" "[winlog][event_data][Hashes]" => "file.hashes"
"[winlog][event_data][TargetFilename]" => "file.target_name"
"[winlog][event_data][CreationUtcTime]" => "file.creation_time"
"[winlog][event_data][ProcessId]" => "process.id"
"[winlog][event_data][ParentProcessId]" => "process.parent.id" } }
if [file][hashes] { ruby { code => ' hashes =
event.get("[file][hashes]") if hashes hashes.split(",").each do
|hash| key_value = hash.split("=") if key_value.length == 2 end end
end ' } } } if [winlog][provider_name] == "PowerShell" { mutate {
rename => { "[winlog][event_id]" => "event.powershell_id"
"[winlog][event_data][param1]" => "powershell.command"
```

```
"[winlog][event_data][param2]" => "powershell.details"
"[winlog][event_data][param3]" => "powershell.output" } } } if
[host][name] { mutate { rename => { "[host][name]" =>
"system.hostname" "[host][os][name]" => "system.os_name"
"[host][os][version]" => "system.os_version" } } } if
[agent][version] { mutate { rename => { "[agent][version]" =>
"logstash.agent_version" } } } mutate { remove_field => [
"[winlog][provider_guid]", "[winlog][record_id]",
"[winlog][opcode]", "[winlog][api]" ]

 } } else if "filebeat-gvm" in [tags] {

if [message] =~ "^\{.*\}$" {
  json {
    source => "message"
  }
} else {
  grok {
    match => { "message" => "%{IPORHOST:client_ip} - -
\[%{HTTPDATE:log_timestamp}\] \"%{WORD:http_method}
%{URIPATHPARAM:request} HTTP/%{NUMBER:http_version}\"
%{NUMBER:response_code} (?:%{NUMBER:bytes_sent}|-)
\"%{DATA:referrer}\" \"%{DATA:user_agent}\"" }
  }
}
date {
  match => [ "log_timestamp", "dd/MMM/yyyy:HH:mm:ss Z" ]
  target => "@timestamp"
}
mutate {
  remove_field => ["log_timestamp"]
}


} else if "filebeat-elasticsearch" in [tags] { json { source =>
"message" target => "audit" skip_on_invalid_json => true } mutate {
add_field => { "user" => "%{[audit][user.name]}" "action" =>
"%{[audit][event.action]}" "timestamp" => "%{[audit][timestamp]}" }
} date { match => ["[audit][@timestamp]", "yyyy-MM-
dd'T'HH:mm:ss,SSSZ"] target => "@timestamp" } } else if "filebeat-
WAF" in [tags] { if [log][file][path] == "/var/log/nginx/access.log"
{ mutate { add_tag => [ "nginx-access" ] } grok { match => {
"message" => '%{IPORHOST:remote_ip} - %{DATA:remote_user}
[%{HTTPDATE:timestamp}] "(?:%{WORD:method}
%{URIPATHPARAM:request}(?: HTTP/%{NUMBER:http_version})?)"
%{NUMBER:status} (?:%{NUMBER:bytes}|-) "(?:%{URI:referrer}|-)"
"(?:%{DATA:user_agent}|-)"' } remove_field => [ "event.original" ] }
date { match => [ "timestamp", "dd/MMM/yyyy:HH:mm:ss Z" ] target =>
"@timestamp" } } else if [log][file][path] ==
"/var/log/nginx/error.log" { mutate { add_tag => [ "nginx-error" ] }
grok { match => { "message" => '[%{DAY} %{MONTH} %{MONTHDAY} %{TIME}
%{YEAR}] [%{LOGLEVEL:log_level}]( [%{DATA:module}])?
```

```
%{GREEDYDATA:error_message}' } remove_field => [ "event.original" ]
} date { match => [ "timestamp", "dd/MMM/yyyy:HH:mm:ss Z" ] target
=> "@timestamp" } } else if [log][file][path] ==
"/var/log/modsec/kibana.fri.uniza.sk_audit.log" { mutate { add_tag
=> [ "nginx-audit" ] } grok { match => { "message" => "(?m)---
%{DATA:modsec_id}---A--\[%{HTTPDATE:timestamp}\] %{NUMBER:unique_id}
%{IP:src_ip} %{NUMBER:src_port} %{IP:dst_ip} %{NUMBER:dst_port}.*?--
-%{DATA:modsec_id}---B--%{GREEDYDATA:http_request}---
%{DATA:modsec_id}---F--%{GREEDYDATA:http_response}---
%{DATA:modsec_id}---H--%{GREEDYDATA:modsec_alerts}" } tag_on_failure
=> [ "_modsec_grok_fail" ] } date { match => [ "timestamp",
"dd/MMM/yyyy:HH:mm:ss Z" ] target => "@timestamp" } mutate {
remove_field => [ "event.original" ] } } } else if "filebeat-
threatintel" in [tags] { date { match => ["[audit][@timestamp]",
"yyyy-MM-dd'T'HH:mm:ss,SSSZ"] target => "@timestamp" } if
[fileset][name] == "abuseurl" { json { source => "message" target =>
"abuseurl" } mutate { add_field => { "threat_type" =>
"%{[abuseurl][threat]}" "url" => "%{[abuseurl][url]}" "ip" =>
"%{[abuseurl][host]}" "reporter" => "%{[abuseurl][reporter]}"
"malware_tag" => "%{[abuseurl][tags][0]}" "urlhaus_ref" =>
"%{[abuseurl][urlhaus_reference]}" } } } else if [fileset][name] ==
"abusemalware" { json { source => "message" target => "abusemalware"
} mutate { add_field => { "sha256" =>
"%{[abusemalware][sha256_hash]}" "md5" =>
"%{[abusemalware][md5_hash]}" "signature" =>
"%{[abusemalware][signature]}" "vt_result" =>
"%{[abusemalware][virustotal][result]}" "vt_percent" =>
"%{[abusemalware][virustotal][percent]}" "download_url" =>
"%{[abusemalware][urlhaus_download]}" } } } else if [fileset][name]
== "otx" { json { source => "message" target => "otx" } mutate {
add_field => { "indicator" => "%{[otx][indicator]}" "indicator_type"
=> "%{[otx][type]}" "description" => "%{[otx][description]}" } } }

 } else if "arkimecapture-moloch" in [tags] { grok { match => {
"message" => "%{MONTH:log_month} %{MONTHDAY:log_day}
%{TIME:log_time} %{GREEDYDATA:rest_of_message}" } } mutate {
add_field => { "log_timestamp" => "%{log_month} %{log_day}
%{log_time}" } } date { match => [ "log_timestamp", "MMM dd
HH:mm:ss" ] target => "@timestamp" timezone => "Europe/Bratislava" }
grok { match => { "message" => "drop: %{NUMBER:drop_count:int}
(%{NUMBER:drop_percent:float})" } } grok { match => { "message" =>
"recv: %{NUMBER:recv_count:int}" } } } }
```

2.1.3.3 Ukážka konfiguračného súboru output.conf pre nasadený Logstash

```
output {
  if "auditbeat-studenti" in [tags] {
    elasticsearch {
      hosts => ["https://192.168.1.178:9200"]
      user => "logstash_internal"
      password => "${ES_psswd}"
      ssl_enabled => true
      ssl_certificate_authorities => "/etc/logstash/certs/ca.crt"
```

```
    ilm_enabled => true
    ilm_rollover_alias => "auditbeat-logs-studenti"
    ilm_pattern => "{now/d}-000001"
    ilm_policy => "auditbeat-logs-ILM"
    manage_template => false
  }
}
else if "suricata-moloch" in [tags] {
  elasticsearch {
    hosts => ["https://192.168.1.178:9200"]
    user => "logstash_internal"
    password => "${ES_psswd}"
    ssl_enabled => true
    ssl_certificate_authorities => "/etc/logstash/certs/ca.crt"
    ilm_enabled => true
    ilm_rollover_alias => "suricata-moloch"
    ilm_pattern => "{now/d}-000001"
    ilm_policy => "suricata-moloch-ILM"
    manage_template => false
  }
}
else if "syslog-fortigate" in [log_type] {
  elasticsearch {
    hosts => ["https://192.168.1.178:9200"]
    user => "logstash_internal"
    password => "${ES_psswd}"
    ssl_enabled => true
    ssl_certificate_authorities => "/etc/logstash/certs/ca.crt"
    ilm_enabled => true
    ilm_rollover_alias => "syslog-fortigate"
    ilm_pattern => "{now/d}-000001"
    ilm_policy => "syslog-fortigate-ILM"
    manage_template => false
  }
}
else if "auditbeat-admin" in [tags] {
  elasticsearch {
    hosts => ["https://192.168.1.178:9200"]
    user => "logstash_internal"
    password => "${ES_psswd}"
    ssl_enabled => true
    ssl_certificate_authorities => "/etc/logstash/certs/ca.crt"
    ilm_enabled => true
    ilm_rollover_alias => "auditbeat-logs-admin"
    ilm_pattern => "{now/d}-000001"
    ilm_policy => "auditbeat-logs-ILM"
    manage_template => false
  }
}
else if "suricata-suricata2" in [tags] {
  elasticsearch {
    hosts => ["https://192.168.1.178:9200"]
```

```
    user => "logstash_internal"
    password => "${ES_psswd}"
    ssl_enabled => true
    ssl_certificate_authorities => "/etc/logstash/certs/ca.crt"
    ilm_enabled => true
    ilm_rollover_alias => "suricata-suricata2"
    ilm_pattern => "{now/d}-000001"
    ilm_policy => "suricata-suricata2-ILM"
    manage_template => false
  }
}
else if "auditbeat-testing" in [tags] {
  elasticsearch {
    hosts => ["https://192.168.1.178:9200"]
    user => "logstash_internal"
    password => "${ES_psswd}"
    ssl_enabled => true
    ssl_certificate_authorities => "/etc/logstash/certs/ca.crt"
    index => "auditbeat-testing"
    manage_template => false
  }
}
else if "filebeat-gvm" in [tags] {
  elasticsearch {
    hosts => ["https://192.168.1.178:9200"]
    user => "logstash_internal"
    password => "${ES_psswd}"
    ssl_enabled => true
    ssl_certificate_authorities => "/etc/logstash/certs/ca.crt"
    ilm_enabled => true
    ilm_rollover_alias => "gvm-greenbone"
    ilm_pattern => "{now/d}-000001"
    ilm_policy => "gvm-greenbone-ILM"
    manage_template => false
  }
}
else if "auditbeat-elk" in [tags] {
  elasticsearch {
    hosts => ["https://192.168.1.178:9200"]
    user => "logstash_internal"
    password => "${ES_psswd}"
    ssl_enabled => true
    ssl_certificate_authorities => "/etc/logstash/certs/ca.crt"
    ilm_enabled => true
    ilm_rollover_alias => "auditbeat-logs-elk"
    ilm_pattern => "{now/d}-000001"
    ilm_policy => "auditbeat-logs-ILM"
    manage_template => false
  }
}
else if "filebeat-elasticsearch" in [tags] {
  elasticsearch {
```

```
    hosts => ["https://192.168.1.178:9200"]
    user => "logstash_internal"
    password => "${ES_psswd}"
    ssl_enabled => true
    ssl_certificate_authorities => "/etc/logstash/certs/ca.crt"
    ilm_enabled => true
    ilm_rollover_alias => "elasticsearch-elk-elasticsearch"
    ilm_pattern => "{now/d}-000001"
    ilm_policy => "elasticsearch-elk-elasticsearch-ILM"
    manage_template => false
  }
}
else if "filebeat-WAF" in [tags] {
  elasticsearch {
    hosts => ["https://192.168.1.178:9200"]
    user => "logstash_internal"
    password => "${ES_psswd}"
    ssl_enabled => true
    ssl_certificate_authorities => "/etc/logstash/certs/ca.crt"
    ilm_enabled => true
    ilm_rollover_alias => "waf-filebeat"
    ilm_pattern => "{now/d}-000001"
    ilm_policy => "waf-filebeat-ILM"
    manage_template => false
  }
}
else if "filebeat-threatintel" in [tags] {
  elasticsearch {
    hosts => ["https://192.168.1.178:9200"]
    user => "logstash_internal"
    password => "${ES_psswd}"
    ssl_enabled => true
    ssl_certificate_authorities => "/etc/logstash/certs/ca.crt"
    ilm_enabled => true
    ilm_rollover_alias => "threatintel-filebeat"
    ilm_pattern => "{now/d}-000001"
    ilm_policy => "threatintel-filebeat-ILM"
    manage_template => false
  }
}
else if "arkimecapture-moloch" in [tags] {
  elasticsearch {
    hosts => ["https://192.168.1.178:9200"]
    user => "logstash_internal"
    password => "${ES_psswd}"
    ssl_enabled => true
    ssl_certificate_authorities => "/etc/logstash/certs/ca.crt"
    ilm_enabled => true
    ilm_rollover_alias => "arkimecapture-moloch"
    ilm_pattern => "{now/d}-000001"
    ilm_policy => "arkimecapture-moloch-ILM"
    manage_template => false
```

```
        }
    }
}
```

2.2 Shuffle

Shuffle Automation je v rámci PoC riešenia INNOSOC využitý pre proces automatizovaného vykonávania procesov a transformácie dát medzi rôznymi bezpečnostnými nástrojmi a systémami (SOAR, Security Automation, Orchestration and Response). V našom prípade ide o orchestráciu dátových tokov, ktorá zabezpečuje, že údaje z jedného systému sú automaticky spracované, transformované a presmerované do ďalších systémov bez ľudskej intervencie.

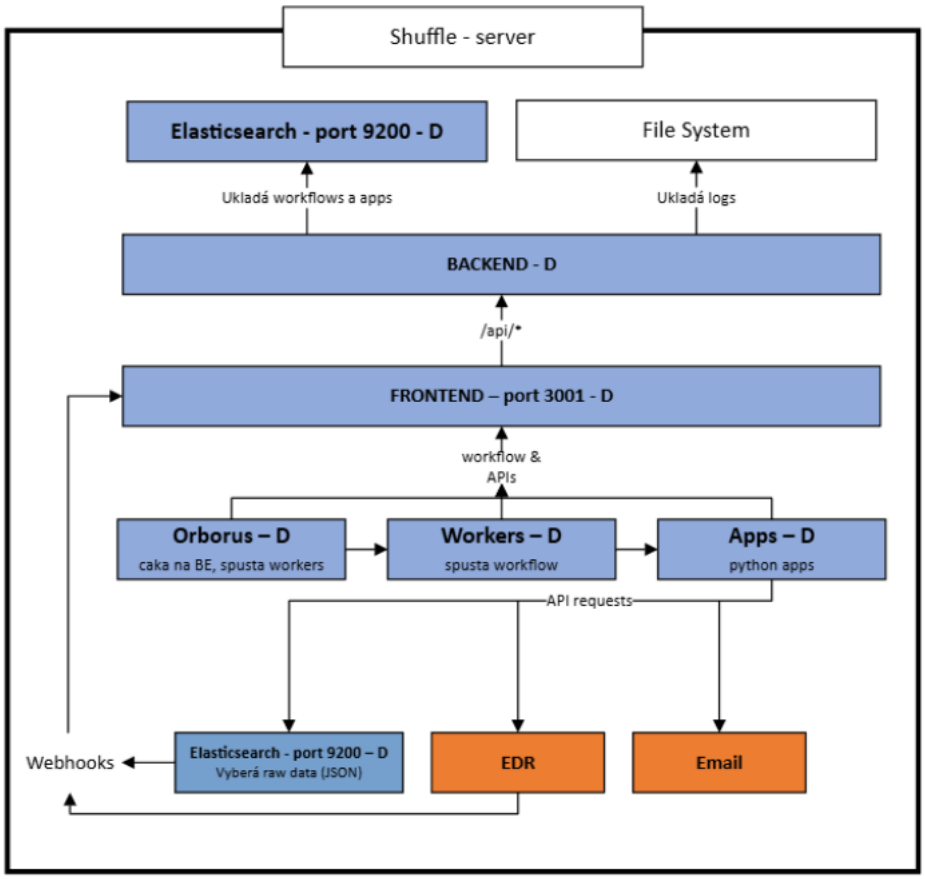
2.2.1 Model nasadenia Shuffle

Riešenie beží v Docker kontajneroch, pričom backend je postavený na OpenSearch, ktorý slúži ako úložisko pre workflows a aplikácie v pythone. Webové rozhranie je prístupné na porte 3001.

Technická špecifikácia nasadenia Shuffle:

- Prostredie – OpenStack
- Meno inštancie – shuffe
- IP adresa – 192.168.4.188
- vCPU – 4
- RAM – 16 GB
- Disk – 80 GB
- OS – Ubuntu 24.04

Softvér Shuffle je založený na dockeri. To znamená, že k tomu aby vedel plne fungovať musí mať systém, na ktorom bude nasadený na najaktuálnejšiu verziu dockera. Spolu s inštaláciou Shuffle sú 2 súbory, docker-compose a súbor .env. Oba súbory slúžia na konfiguráciu portov, port forwardingu, backendu a frontendu. K tomu aby Shuffle vedel plne fungovať, stačí ho zapnúť pomocou príkazu sudo docker-compose up –d.



Obrázok 1 Architektúra komponentov Shuffle

Obrázok znázorňuje architektúru SOAR platformy Shuffle na serveri a tok spracovania dát. Frontend beží na porte 3001 a slúži ako webové rozhranie, cez ktoré používateľ pracuje s workflowmi a API. Požiadavky z frontendu sú smerované na backend, ktorý zabezpečuje logiku aplikácie a komunikáciu s databázou Elasticsearch, kde sa ukladajú workflows a aplikácie, a so súborovým systémom, v ktorom sa ukladajú logy. V runtime prostredí sa nachádzajú komponenty Orborus, Workers a Apps, pričom Orborus dohliada na backend a spúšťa workers, ktoré vykonávajú jednotlivé kroky workflowov. Apps predstavujú skripty, najčastejšie v jazyku Python, ktoré komunikujú s externými službami. Prostredníctvom API požiadaviek a webhookov je Shuffle prepojený s externými systémami, ako sú Elasticsearch, EDR alebo e-mailové služby.

2.2.2 Základné integračné väzby komponentu Shuffle v PoC

Aktuálne prepojené a otestované služby pomocou Shuffle:

- ELK – vieme si vyberať dáta pomocou volania API, vyhľadávať indexy a vytvárať nové indexy. Je ale nutné mať v iptables na zariadení kde beží ELK, povolené IP zariadenia na ktorom beží Shuffle.
- Discord – vieme posielať notifikácie vo formáte JSON pomocou webhooku ktorý sa vytvára v skupine na Discorde.
- OpenCTI – vieme vytvárať pomocou API nové indikátory, tieto vytvárame ako JSON a posielame pomocou API volania.
- DFIR IRIS – vieme pomocou API, vytvárať nové eventy, ktoré taktiež vieme poslať ako formát JSON.

2.3 Wazuh

Wazuh je v rámci PoC riešenia INNOSOC využitý ako centrálna platforma pre zber, koreláciu a analýzu bezpečnostných udalostí s funkcionalitou EDR.

2.3.1 Model nasadenia komponentu Wazuh

Architektúra nášho riešenia je založená na nasadení univerzálneho Wazuh-agenta na cieľové systémy a troch kľúčových centrálnych komponentov:

- Wazuh-manager (server): Zabezpečuje analýzu dát, správu agentov a detekciu hrozieb.
- Wazuh-indexer (databáza): Vysoko výkonný full-textový vyhľadávací a analytický nástroj slúžiaci na indexovanie a ukladanie bezpečnostných záznamov.
- Wazuh-dashboard: Webové používateľské rozhranie určené na vizualizáciu dát, analýzu incidentov a konfiguráciu systému

Centrálne komponenty sú nasadené na virtuálnom stroji (VM) v prostredí XenOrchestra. VM má k dispozícii: 8 virtuálnych CPU, 8GB RAM pamäte a 250GB úložiska.

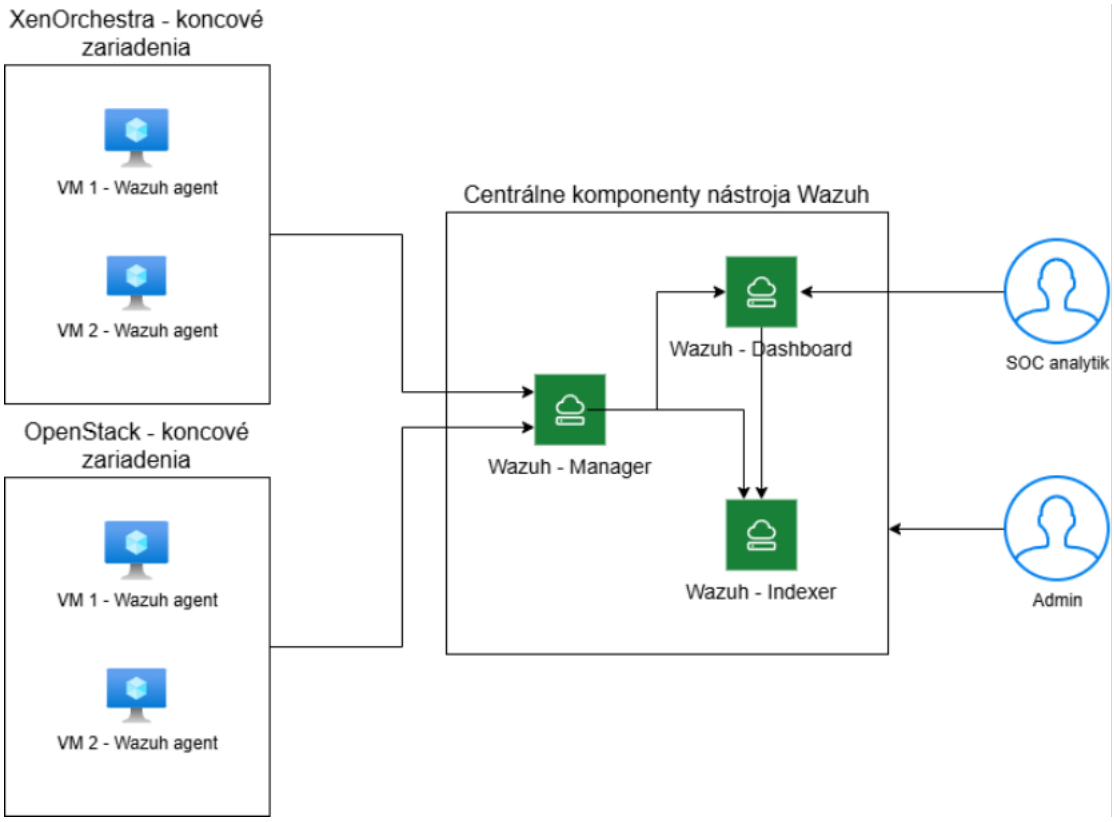
2.3.2 Monitorované prostredia nástrojom Wazuh

Monitorujeme prostredia dvoch tenatov. Prvým sú cloudové prostredia na Katedry informačných sietí na Fakulte riadenia a informatiky XenOrchestra a OpenStack. Druhé je prostredie vybranej ZŠ v Žiline.

2.3.2.1 Prostredie Katedry informačných sietí FRI UNIZA

V podmienkach KIS je systém konfigurovaný ako All-in-one inštalácia, pričom všetky kľúčové komponenty platformy Wazuh sú integrované do jedného virtuálneho stroja (VM). Toto riešenie zabezpečuje centralizovaný zber a analýzu dát z monitorovaných uzlov nachádzajúcich sa v prostrediach XenOrchestra a OpenStack. Grafické znázornenie výslednej sieťovej architektúry a hierarchie komponentov je ilustrované na obrázku nižšie.

Keďže sa jedná o spôsob nasadenia All-in-one tak všetky centrálne komponenty majú rovnakú IP adresu: 192.168.1.184. V prípade inštalácie Single alebo multi-node je potrebné aby mal každý komponent vlastnú IP adresu (keďže každý komponent bude na samostatnom servery alebo VM).



2.3.2.2 *Monitorovanie vybranej ZŠ v Žiline*

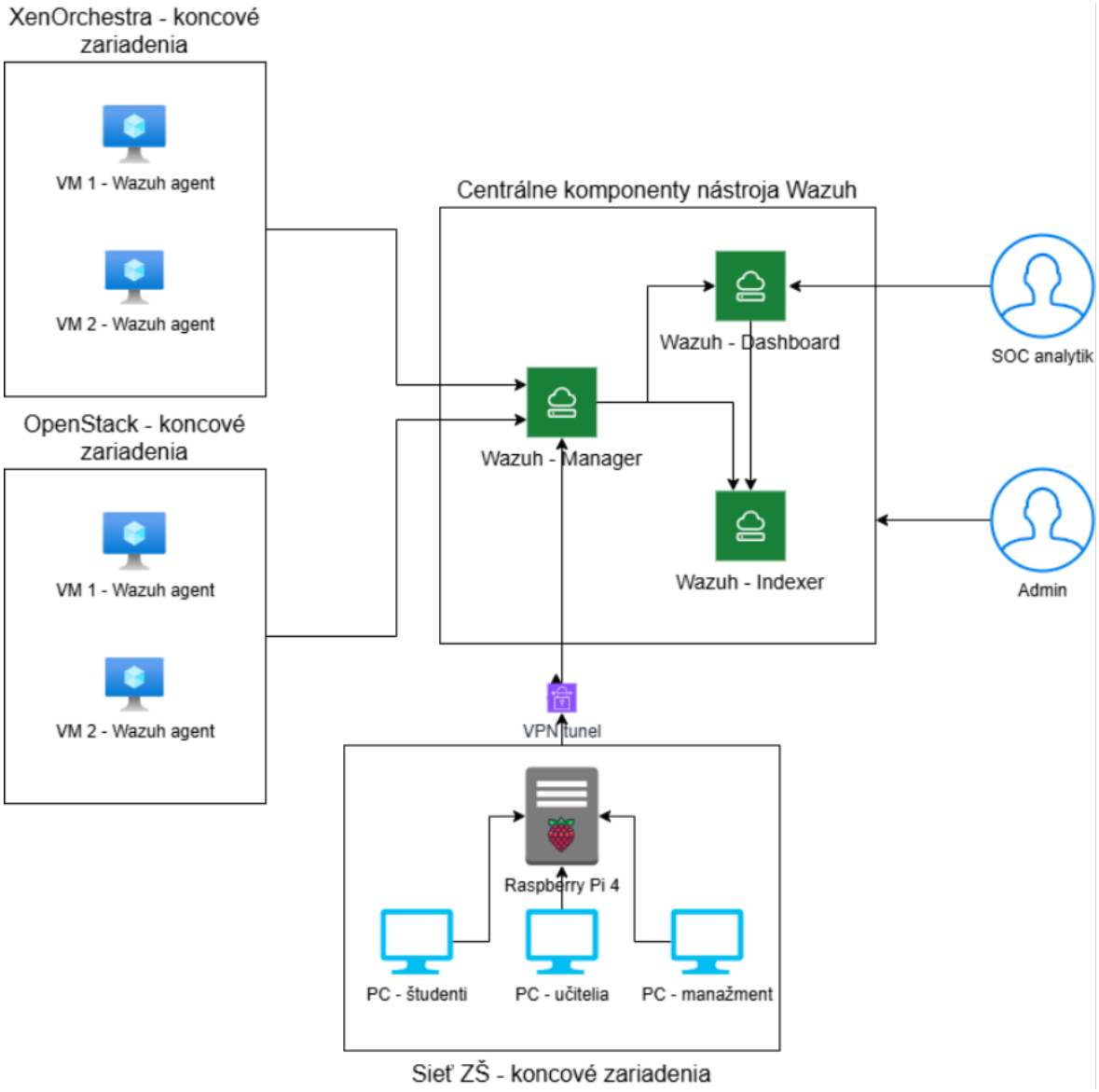
Druhá inštancia monitorovacieho systému je implementovaná v infraštruktúre vybranej základnej školy (ZŠ) v Žiline. Pôvodná architektúra v tomto prostredí pozostávala z lokálneho servera, na ktorom boli nasadené základné komponenty platformy Wazuh. Tento server zabezpečoval zber a ukladanie dát z koncových zariadení v rámci lokálnej siete ZŠ, pričom vzdialený prístup k rozhraniu Wazuh Dashboard bol realizovaný prostredníctvom VPN technológie ZeroTier.

Z dôvodu neefektívnej správy a nutnosti redundantnej konfigurácie dvoch samostatných manažérov došlo k revízii architektúry. Rozhodlo sa o centralizácii hlavných komponentov do prostredia XenOrchestra na KIS FRI. Dáta z koncových zariadení umiestnených v ZŠ sú po novom presmerované a spracovávané centrálne na KIS.

Prepojenie oboch lokalít je technicky riešené nasledovne:

- **Strana KIS:** Na centrálnom serveri sú okrem manažéra Wazuh prevádzkované aj serverové komponenty VPN Wireguard.
- **Strana ZŠ:** V lokálnej sieti je nasadené zariadenie Raspberry Pi 4, ktoré plní úlohu VPN klienta.

- **Sieťová konektivita:** Medzi lokalitami je vytvorený funkčný VPN tunel s adresným rozsahom 10.0.0.0/24.
- **Smerovanie (Routing):** Na lokálnom routeri v ZŠ je definovaná statická cesta (route), ktorá smeruje všetku prevádzku určenú pre sieť `10.0.0.0/24` cez bránu Raspberry Pi.



- **Wazuh-agenti** nainštalovaný na koncových zariadeniach v ZŠ odosielajú zozbierané dáta priamo na Wazuh-manažéra v sieti KIS prostredníctvom zabezpečeného tunela.

2.3.3 Nasadené a aktivované moduly systému Wazuh

V úvodnej fáze boli aktivované a konfigurované natívne moduly systému Wazuh, ktoré tvoria základný rámec pre monitorovanie bezpečnosti a dodržiavanie súladu (compliance). Nižšie sú detailne špecifikované vybrané funkcie:

2.3.3.1 *File Integrity Monitoring (FIM)*

Tento modul monitoruje zmeny v kritických systémových súboroch a adresároch. Funguje na princípe porovnávania kontrolných súčtov (checksums) a atribútov súborov v čase.

- **Mechanizmus:** Pri inicializácii vytvorí agent databázu hashov vybraných súborov. Pri každej zmene (vytvorenie, modifikácia, vymazanie) odošle upozornenie manažérovi.
- **Využitie:** Detekcia neoprávnených zmien v konfiguráciách alebo injekcia malvéru.

2.3.3.2 *Rootcheck (Rootkit Detection)*

Modul zameraný na detekciu skrytých hrozieb, ako sú rootkity, trójske kone alebo neoprávnené modifikácie jadra operačného systému.

- **Mechanizmus:** Vykonáva hĺbkovú kontrolu anomálií v systéme, porovnáva výstupy systémových volaní s priamym čítaním pôvodných (tzv. raw) dát a hľadá skryté procesy alebo porty.

2.3.3.3 *Vulnerability Detection (Detekcia zraniteľností)*

Tento modul automaticky identifikuje slabé miesta v operačnom systéme a nainštalovaných aplikáciách.

- **Mechanizmus:** Wazuh agent zhromažďuje zoznam nainštalovaných balíkov, ktorý následne manažér porovnáva s globálnymi databázami zraniteľností (**CVE** – Common Vulnerabilities and Exposures) v NVD - National Vulnerability Database.

2.3.3.4 *Policy monitoring & SCA (Security Configuration Assesment)*

Slúži na overenie, či je systém nakonfigurovaný v súlade s bezpečnostnými štandardmi (napr. CIS Benchmarks).

- **Mechanizmus:** Modul spúšťa skripty, ktoré kontrolujú nastavenia systému (napr. silu hesiel, zakázané protokoly). Výsledkom je skóre súladu a zoznam odporúčaní na nápravu.

2.3.3.5 *System Inventory (Systémová inventarizácia)*

Poskytuje komplexný prehľad o hardvérových a softvérových vlastnostiach monitorovaného endpointu.

Mechanizmus: Zhromažďuje údaje o sieťových rozhraniach, bežiacich procesoch, nainštalovaných balíkoch a verziách OS. Tieto dáta sú dostupné v reálnom čase cez wazuh-dashboard.

2.3.4 Základné integračné väzby - posielanie notifikácií

Ekosystém nástroja Wazuh ponúka široké spektrum integračných možností pre automatizované zasielanie výstrah prostredníctvom externých komunikačných kanálov a nástrojov tretích strán.

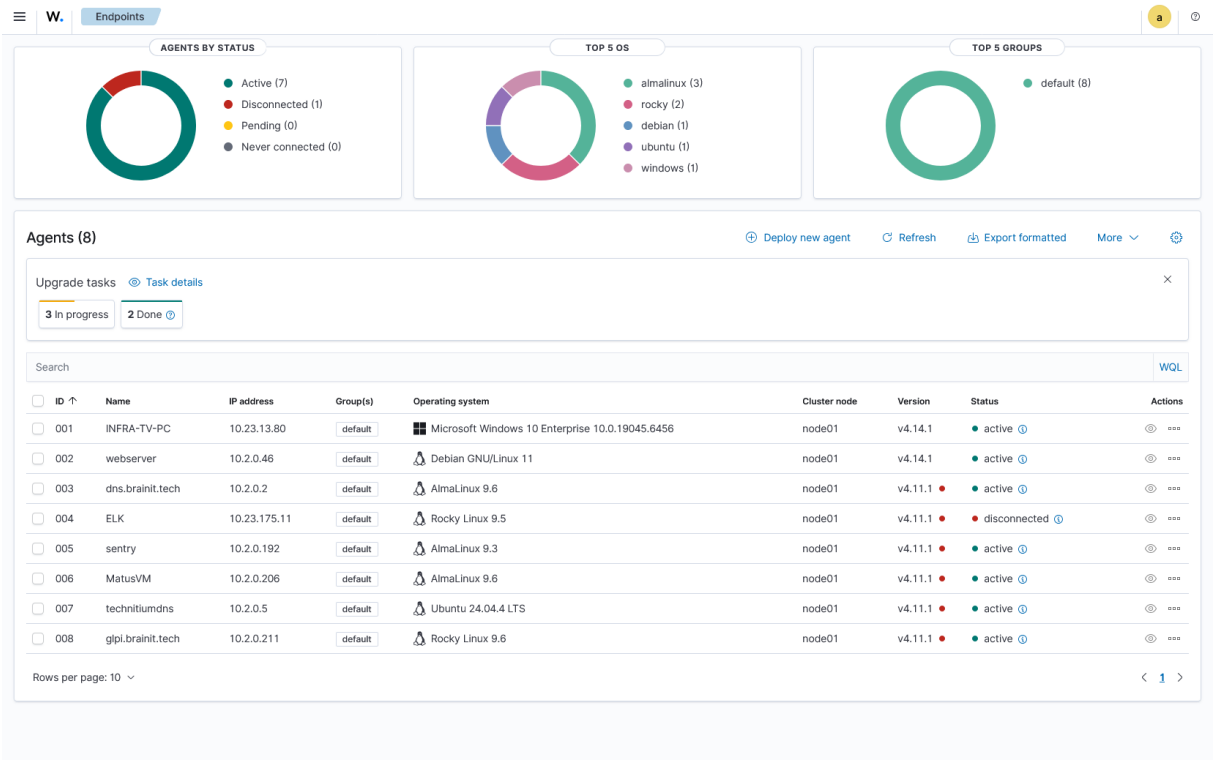
Posielanie notifikácií do externých platforiem (Slack, MS Teams, PagerDuty a pod.) je vo Wazuhu riešené cez modul Integrations. Integrácia je realizovaná prostredníctvom aplikačného rozhrania Webhook, pričom systém umožňuje granulárne filtrovanie odosielaných udalostí na základe ich závažnosti. Tento proces umožňuje transformovať interné výstrahy (Alerts) na okamžité správy pre bezpečnostný tím, čím sa výrazne skracuje reakčný čas (MTTR – Mean Time to Respond).

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.

Hlavným dôvodom výberu bola skutočnosť, že táto platforma je plne integrovaná v existujúcej digitálnej infraštruktúre univerzity. Využitie ekosystému Microsoft 365 nám prináša viaceré výhody:

- **Efektivita nasadenia:** Odpadá nutnosť budovania novej komunikačnej infraštruktúry alebo správy ďalších používateľských účtov.
- **Konektivita:** Integrácia prebieha v rámci overeného a zabezpečeného prostredia univerzity.
- **Dostupnosť:** Zabezpečuje okamžitý prístup k notifikáciám pre všetkých členov tímu bez potreby inštalácie dodatočného softvéru tretích strán.

Ukážka z dashboardu nástroja Wazuh:

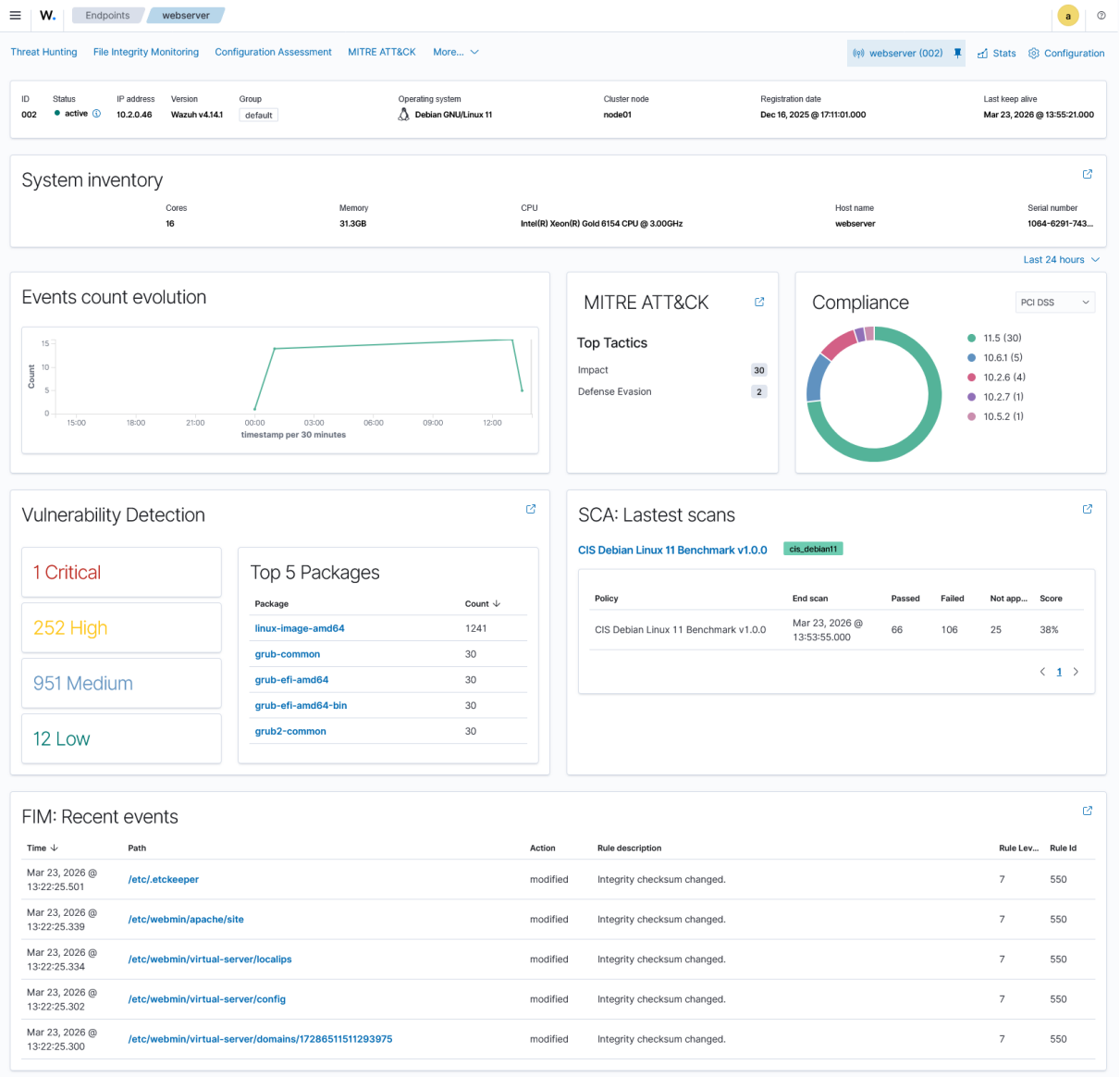


Zobrazené rozhranie dopĺňa integračný scenár zasielania notifikácií tým, že poskytuje prehľad o zdrojových endpointoch, z ktorých sú bezpečnostné udalosti generované. Dashboard zobrazuje stav agentov, ich dostupnosť a základné charakteristiky, ktoré vstupujú do procesu detekcie a následného odosielania alertov.

Prehľad stavu agentov umožňuje identifikovať, či sú všetky relevantné koncové body aktívne a správne pripojené, čo je predpokladom pre spoľahlivé generovanie notifikácií. Neaktívne alebo odpojené zariadenia môžu predstavovať výpadok v pokrytí detekcie a tým aj absenciu notifikácií pre bezpečnostný tím.

Tabuľkový prehľad agentov zároveň poskytuje kontextové informácie, ktoré sa prenášajú aj do notifikácií, ako napríklad identifikácia agenta, operačný systém alebo verzia. Tieto údaje sú následne súčasťou alertov odosielaných cez webhook do MS Teams a umožňujú rýchlejšie vyhodnotenie udalosti.

Rozhranie tak predstavuje vstupnú vrstvu celého notifikačného procesu, kde kvalita a dostupnosť dát z agentov priamo ovplyvňuje presnosť a spoľahlivosť zasielaných bezpečnostných upozornení.



Obrázok zobrazuje detailný pohľad na konkrétny endpoint v rámci modulu **Threat Hunting / Endpoint Detail** v prostredí Wazuh. Rozhranie poskytuje konsolidovaný prehľad telemetrie, bezpečnostných udalostí a konfiguračného stavu zariadenia.

V hornej časti sú uvedené identifikačné a prevádzkové metadáta endpointu, vrátane IP adresy, verzie agenta, operačného systému, cluster node a časových údajov poslednej komunikácie. Tieto informácie definujú kontext zdroja dát a jeho dostupnosť v rámci monitorovania.

Sekcia *System inventory* poskytuje základné hardvérové a systémové parametre (CPU, pamäť, hostname), ktoré sú využiteľné pri korelácii výkonových a bezpečnostných udalostí.

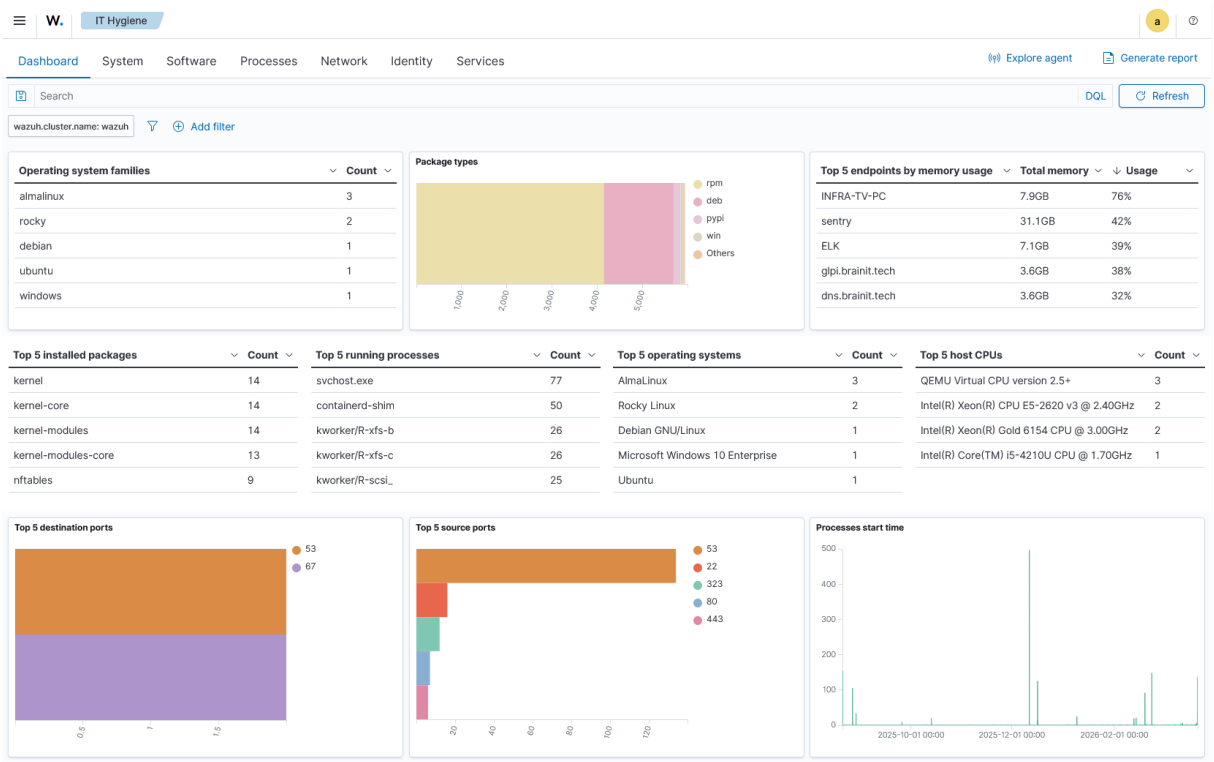
Časová séria *Events count evolution* zobrazuje vývoj počtu udalostí v čase, čo umožňuje identifikovať anomálie v správaní systému alebo náhle nárasty aktivity. Sekcia *MITRE ATT&CK* mapuje detegované aktivity na konkrétne taktiky (napr. Defense Evasion), čím poskytuje štandardizovaný rámec pre interpretáciu hrozieb.

Modul *Compliance* zobrazuje stav súladu systému s definovanými politikami a benchmarkmi, zatiaľ čo sekcia *Vulnerability Detection* kvantifikuje zistené zraniteľnosti podľa závažnosti. Prehľad najčastejšie sa vyskytujúcich balíkov umožňuje identifikovať potenciálne rizikové komponenty v systéme.

Časť *SCA (Security Configuration Assessment)* poskytuje výsledky posledných kontrol konfigurácie podľa definovaných štandardov (napr. CIS benchmark), vrátane úspešnosti jednotlivých kontrol a celkového skóre.

Spodná sekcia *FIM (File Integrity Monitoring)* eviduje nedávne zmeny v súborovom systéme, vrátane upravených súborov, času zmeny a priradených detekčných pravidiel. Tieto udalosti sú kľúčové pre detekciu neautorizovaných zásahov do systému.

Z architektonického pohľadu toto rozhranie predstavuje korelačný bod nad endpoint telemetriou, kde sú agregované výstupy z viacerých detekčných mechanizmov (FIM, vulnerability scanning, SCA, log analysis) do jednotného kontextu konkrétneho zariadenia. Tento prístup umožňuje efektívnu triáž, analýzu incidentov a rozhodovanie o reakčných opatreniach v rámci SOC.



Obrázok zobrazuje dashboard modulu **IT Hygiene** v prostredí Wazuh, ktorý poskytuje agregovaný pohľad na stav endpointov z hľadiska softvérového a prevádzkového kontextu. Tento modul rozširuje základnú telemetriu o informácie využiteľné pre bezpečnostnú analýzu a správu zraniteľností.

Horná časť dashboardu obsahuje sumarizačné metriky nad inventárom systémov. Zobrazené sú distribúcie operačných systémov, typov balíkov a využitia pamäte na jednotlivých endpointoch. Tieto údaje sú odvodené zo zbieranej telemetrie agentov a slúžia

ako vstup pre hodnotenie expozície a identifikáciu potenciálne rizikových systémov (napr. zastarané OS alebo neštandardné konfigurácie).

Stredná časť poskytuje detailnejší pohľad na softvérový stav endpointov, konkrétne:

- najčastejšie inštalované balíky (napr. kernel komponenty),
- aktívne bežiace procesy (napr. kontajnerové runtime alebo systémové služby),
- rozdelenie endpointov podľa operačných systémov,
- identifikáciu najviac zaťažených systémov z pohľadu CPU.

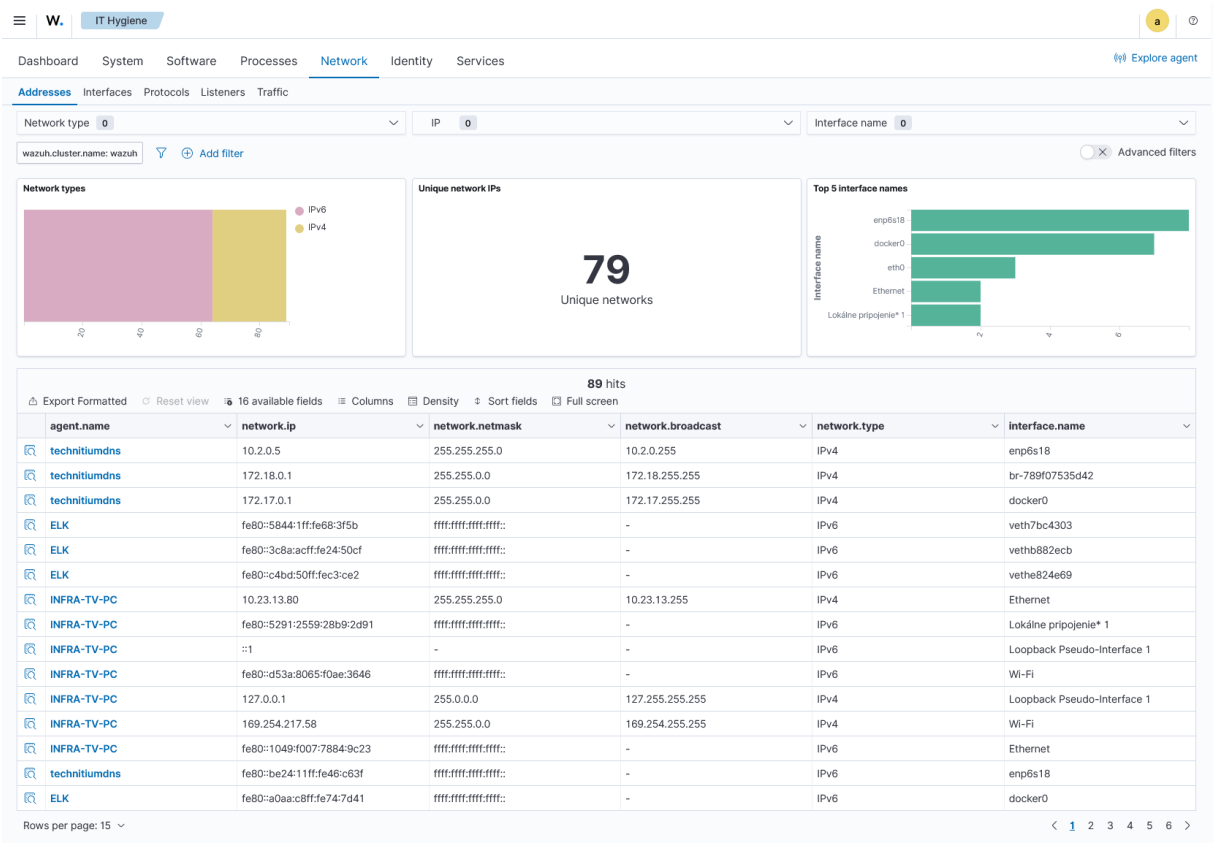
Tieto informácie umožňujú koreláciu medzi systémovým správaním a bezpečnostnými udalosťami, napríklad identifikáciu anomálnych procesov alebo neštandardného využitia zdrojov.

Spodná časť dashboardu sa zameriava na sieťový a časový kontext:

- prehľad najčastejšie využívaných zdrojových a cieľových portov,
- časové rozloženie spúšťania procesov.

Tieto metriky poskytujú základ pre detekciu odchýlok v sieťovej komunikácii alebo správaní procesov, čo je relevantné napríklad pri identifikácii laterálneho pohybu alebo neautorizovaných služieb.

Z architektonického pohľadu tento dashboard reprezentuje analytickú vrstvu nad endpoint telemetriou, kde dochádza k agregácii a vizualizácii dát získaných z agentov. Výstupy slúžia ako podklad pre detekčné pravidlá, threat hunting a prioritizáciu bezpečnostných opatrení v rámci SOC.

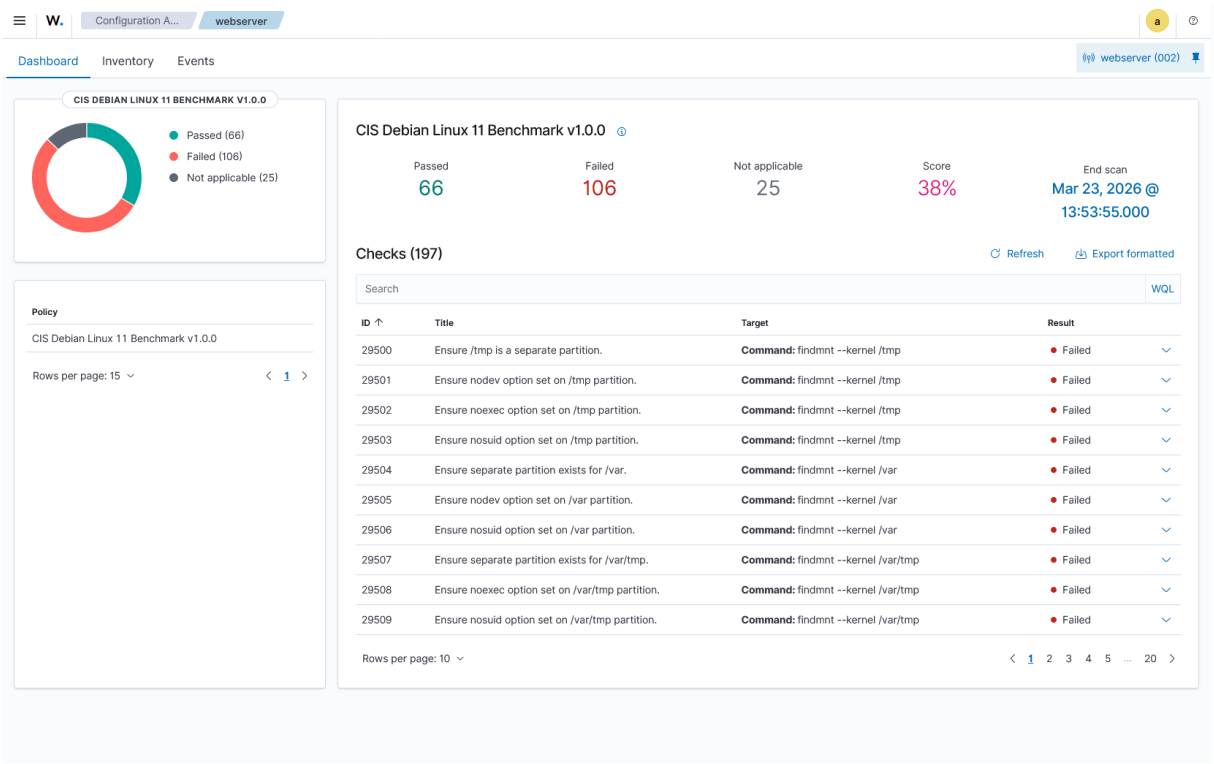


Obrázok zobrazuje dashboard modulu **Network** v rámci Wazuh IT Hygiene, ktorý poskytuje prehľad o sieťovej konfigurácii monitorovaných endpointov.

Horná časť obsahuje agregované metriky nad sieťovými vlastnosťami, konkrétne rozdelenie podľa typu sieťového protokolu (IPv4/IPv6), počet unikátnych sietí a najčastejšie používané sieťové rozhrania. Tieto údaje umožňujú rýchlu identifikáciu štruktúry sieťového prostredia a distribúcie rozhraní naprieč zariadeniami.

Spodná časť obsahuje detailný tabuľkový prehľad sieťových konfigurácií jednotlivých agentov. Pre každý záznam sú uvedené atribúty ako IP adresa, sieťová maska, broadcast adresa, typ siete a názov sieťového rozhrania. Dáta sú zbierané prostredníctvom agentov a reprezentujú aktuálny stav sieťovej konfigurácie endpointov.

Rozhranie umožňuje filtrovanie a analýzu sieťových údajov, čím podporuje identifikáciu neštandardných konfigurácií, duplicitných adries alebo odchýlok v sieťovej topológii. Z pohľadu bezpečnosti tieto informácie slúžia ako vstup pre detekciu anomálií v komunikácii a hodnotenie sieťovej expozície zariadení.



Obrázok zobrazuje výstup modulu **Security Configuration Assessment (SCA)** v prostredí Wazuh, konkrétne hodnotenie endpointu podľa štandardu **CIS Debian Linux 11 Benchmark v1.0.0**.

Horná časť rozhrania poskytuje sumarizačný prehľad výsledkov kontroly konfigurácie. Zobrazený je počet úspešných (Passed), neúspešných (Failed) a neaplikovateľných (Not applicable) kontrol spolu s celkovým skóre súladu (Score). Tieto metriky kvantifikujú úroveň bezpečnostnej konfigurácie systému voči definovanému benchmarku a umožňujú rýchle posúdenie jeho bezpečnostného stavu.

Hlavná časť obsahuje detailný zoznam jednotlivých kontrol (checks), ktoré sú definované v rámci CIS benchmarku. Pre každú kontrolu je uvedený identifikátor, popis požiadavky, cieľový komponent (napr. konkrétny konfiguračný súbor alebo systémový parameter) a výsledok vyhodnotenia. Väčšina kontrol je v stave „Failed“, čo indikuje nesúlad konfigurácie systému s odporúčanými bezpečnostnými praktikami.

Kontroly sú zamerané najmä na hardening operačného systému, napríklad:

- konfiguráciu diskových oddielov (napr. /tmp, /var),
- nastavenie mount parametrov,
- systémové bezpečnostné politiky.

Z architektonického pohľadu tento modul predstavuje vrstvu hodnotenia konfigurácie nad endpoint telemetriou, kde sa systémové nastavenia porovnávajú s definovanými bezpečnostnými štandardmi. Výstupy slúžia ako podklad pre identifikáciu slabých miest, prioritizáciu nápravných opatrení a preukázanie súladu s bezpečnostnými požiadavkami v rámci SOC alebo auditných procesov.

2.4 Greenbone Community edition

2.4.1 Úloha Greenbone Community Edition v PoC

Greenbone Community Edition je v rámci PoC riešenia využitý ako nástroj pre riadenie zraniteľností a identifikáciu bezpečnostných slabín v infraštruktúre. V navrhovanom SOC riešení neplní úlohu detekčného ani reakčného komponentu v reálnom čase, ale predstavuje preventívnu vrstvu, ktorá poskytuje systematický prehľad o zraniteľnostiach aktív. Jeho význam spočíva najmä v tom, že umožňuje pravidelné skenovanie infraštruktúry, identifikáciu slabých miest a poskytovanie podkladov pre riadenie rizík a plánovanie nápravných opatrení.

V rámci PoC je Greenbone chápaný ako nástroj, ktorý dopĺňa detekčné a analytické komponenty (napr. SIEM, EDR) o preventívny bezpečnostný kontext. Výstupy zo skenovania umožňujú identifikovať zraniteľné systémy, priradiť ich ku konkrétnym aktívam a podporiť rozhodovanie o prioritách nápravy. Takýto prístup rozširuje pohľad SOC z reaktívneho spracovania udalostí na proaktívne riadenie bezpečnosti.

Greenbone v SOC poskytuje pohľad na úroveň zraniteľností prostredia v čase. V tomto zmysle predstavuje doplnkový, no dôležitý komponent riešenia, ktorý podporuje prevenciu bezpečnostných incidentov a vytvára základ pre ďalšie integračné scenáre, napríklad prepojenie s ticketing systémami alebo nástrojmi na riadenie rizík.

2.4.2 Nasadenie a základná konfigurácia Greenbone

V rámci PoC riešenia bol Greenbone Community Edition nasadený ako samostatná služba určená pre skenovanie zraniteľností a správu výsledkov. Z pohľadu implementácie je riešenie založené na modulárnej architektúre, ktorá zahŕňa skenovací komponent (OpenVAS Scanner), riadiacu vrstvu (gvmd) a databázové úložisko (PostgreSQL). Takýto prístup umožňuje flexibilné nasadenie a škálovanie podľa rozsahu infraštruktúry.

Základná konfigurácia bola zameraná na sprístupnenie webového rozhrania (GSA) a nastavenie základných skenovacích profilov, cieľov a plánov skenovania. Systém bol pripravený na využívanie aktualizovaného feedu zraniteľností (NVT), ktorý zabezpečuje aktuálnosť detekovaných slabín.

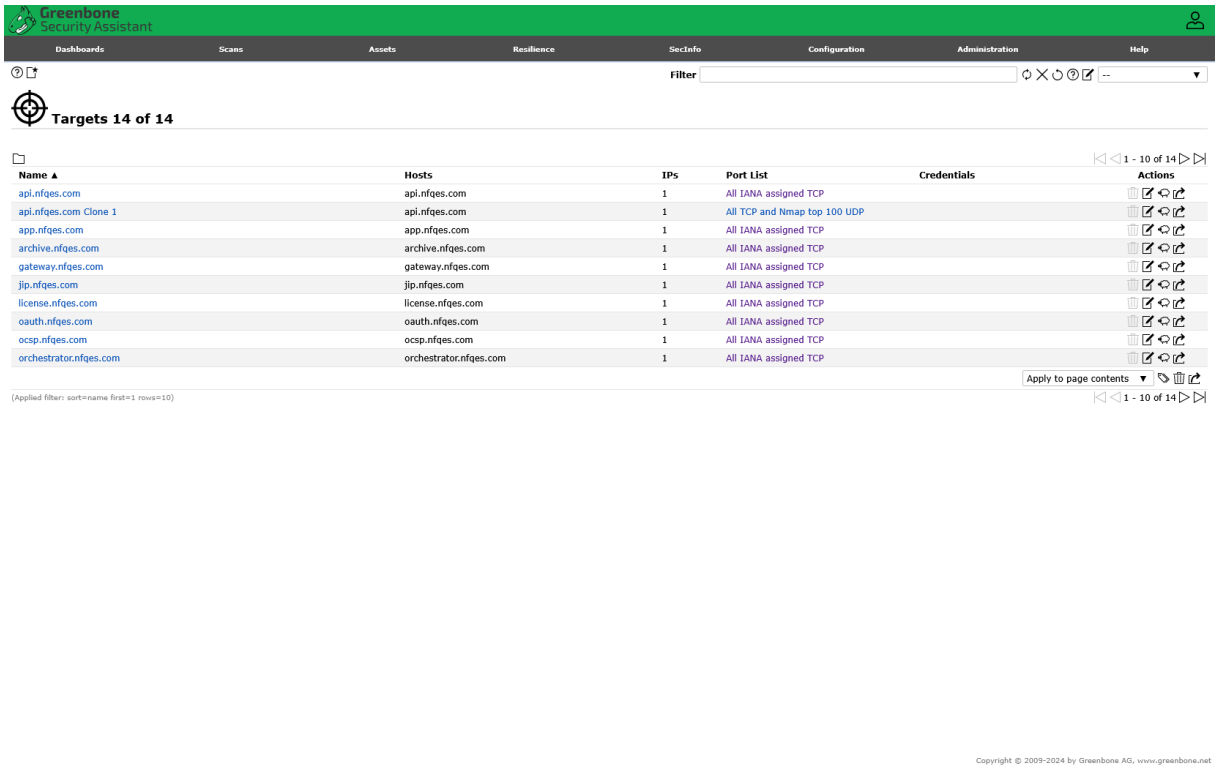
V rámci PoC boli definované základné skenované rozsahy a testované scenáre plánovaného a manuálneho skenovania. Výsledky sú ukladané do databázy a sprístupnené prostredníctvom webového rozhrania, kde je možné ich filtrovať, analyzovať a exportovať.

2.4.3 Využitie Greenbone v rámci PoC

V rámci PoC bol Greenbone využívaný ako nástroj pre identifikáciu zraniteľností a poskytovanie prehľadu o bezpečnostnom stave infraštruktúry. Jeho hlavný prínos spočíval v tom, že umožňoval systematicky odhaľovať slabé miesta systémov a poskytovať podklady pre ich následné riešenie.

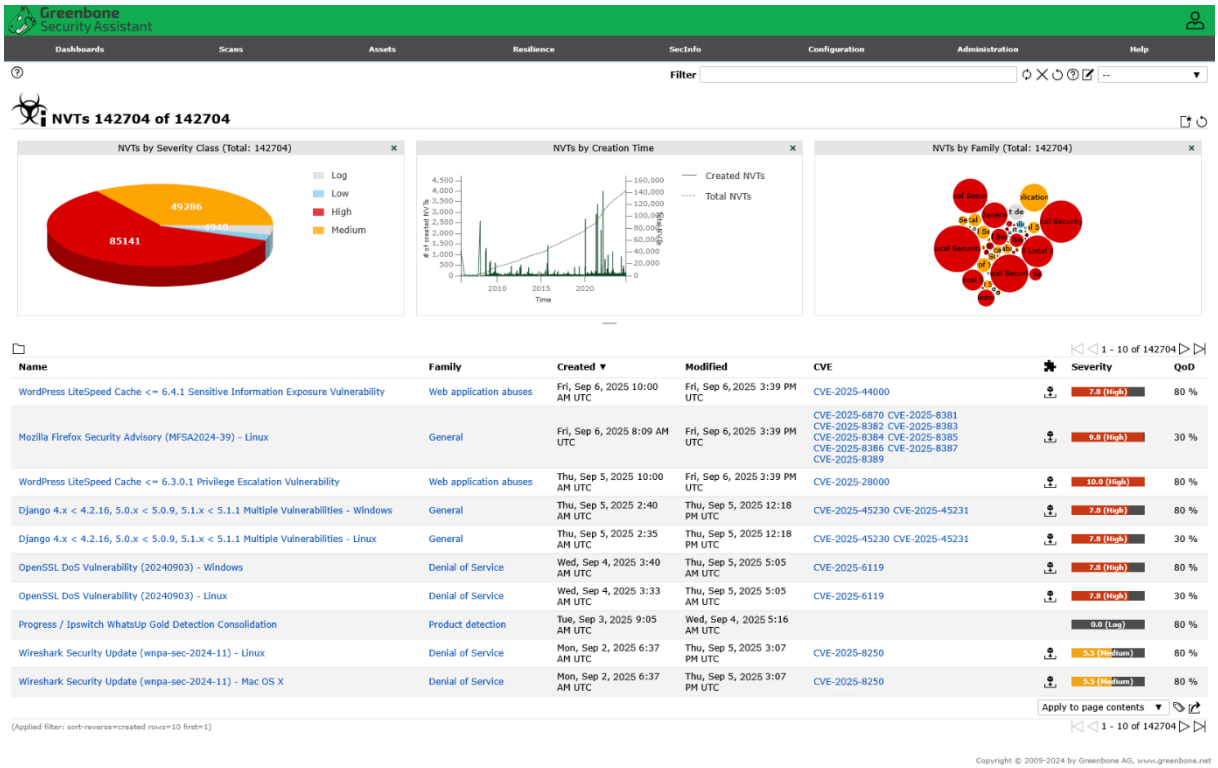
Praktické využitie v tejto etape bolo orientované najmä na vykonávanie základných skenov, analýzu výsledkov a identifikáciu kritických zraniteľností. Greenbone tak neplnil úlohu aktívnej detekcie ani reakcie, ale dopĺňal riešenie o preventívny pohľad na bezpečnosť prostredia.

V tomto rozsahu jeho nasadenie potvrdilo, že ide o vhodný doplnok PoC riešenia a o použiteľný základ pre ďalšie rozšírenie, najmä v oblasti integrácie s procesmi riadenia zraniteľností a incidentov.



Obrázok zobrazuje sekciu **Targets** v nástroji Greenbone Security Assistant, kde sú definované cieľové systémy pre skenovanie zraniteľností.

Tabuľka obsahuje zoznam domén (hostov), ich IP adresy, rozsah skenovaných portov a dostupné akcie správy. Každý záznam predstavuje jeden skenovací cieľ, ktorý určuje, na ktoré systémy a v akom rozsahu bude vykonané bezpečnostné skenovanie.



Obrázok zobrazuje webové rozhranie nástroja **Greenbone Security Assistant (GSA)**, konkrétne prehľad databázy **NVT (Network Vulnerability Tests)**, ktoré predstavujú jednotlivé detekčné testy používané pri skenovaní zraniteľností.

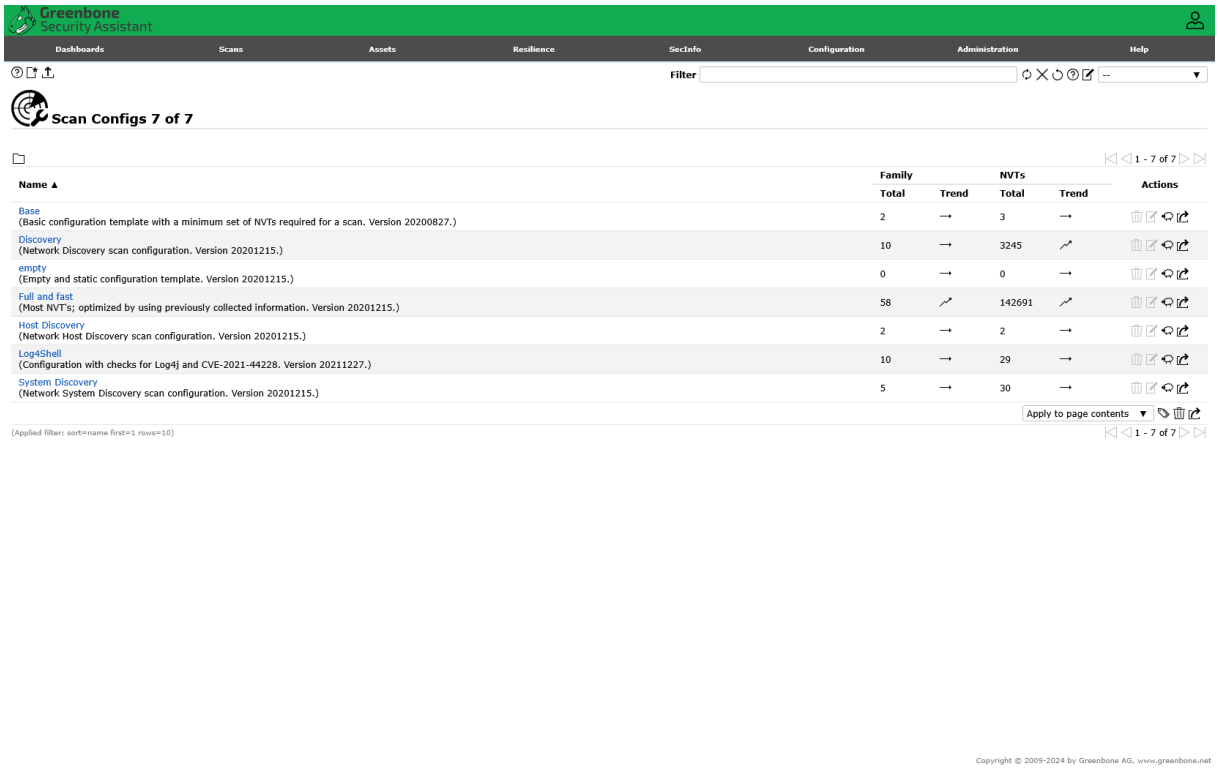
V hornej časti sú vizualizácie poskytujúce agregovaný pohľad na databázu zraniteľností:

- koláčový graf distribúcie NVT podľa závažnosti (Low, Medium, High, Log),
- časový vývoj tvorby NVT testov,
- rozdelenie NVT podľa kategórií (family).

Centrálna časť obsahuje tabuľkový prehľad konkrétnych zraniteľností, kde každý záznam reprezentuje jeden detekčný test. Pre každý test sú uvedené:

- názov a typ zraniteľnosti,
- kategória (family),
- dátum vytvorenia a poslednej úpravy,
- väzba na štandardizované identifikátory (CVE),
- hodnotenie závažnosti (CVSS skóre),
- QoD (Quality of Detection), teda spoľahlivosť detekcie.

Rozhranie umožňuje filtrovanie, triedenie a detailnú analýzu jednotlivých zraniteľností, čím poskytuje analytikom prehľad o pokrytí známych hrozieb v rámci skenovacieho enginu.



Obrázok zobrazuje sekciu **Scan Configs** v nástroji Greenbone Security Assistant, ktorá slúži na správu konfiguračných profilov pre skenovanie zraniteľností.

Tabuľka obsahuje zoznam dostupných skenovacích konfigurácií (napr. *Full and fast*, *Discovery*, *Host Discovery*, *System Discovery*), pričom každá konfigurácia definuje rozsah a typ vykonávaných testov (NVT). Pre jednotlivé konfigurácie sú uvedené:

- stručný popis účelu skenu,
- počet kategórií testov (Family),
- celkový počet použitých NVT testov,
- trend (indikácia zmien alebo aktualizácií),
- akcie pre správu konfigurácie.

Konfigurácie sa líšia najmä rozsahom – od základných alebo rýchlych skenov (Discovery) až po komplexné skenovanie (Full and fast), ktoré využíva rozsiahlu databázu testov. Táto sekcia umožňuje prispôsobiť skenovanie podľa požadovanej hĺbky analýzy, výkonových nárokov a konkrétneho účelu bezpečnostného hodnotenia.

2.4.4 Aktuálny stav a obmedzenia Greenbone v SOC

Greenbone Community Edition bol v rámci PoC úspešne nasadený a základne nakonfigurovaný ako funkčný nástroj pre skenovanie zraniteľností a správu výsledkov. V tomto stave poskytuje použiteľný základ pre identifikáciu bezpečnostných slabín a podporu procesov riadenia zraniteľností. Z pohľadu PoC sa potvrdilo, že ide o vhodný komponent na doplnenie SOC riešenia o preventívny bezpečnostný kontext.

Taktiež je potrebné poukázať na to že NVT feedy a skenovanie z veľkej časti fungujú len na základe korelácii verzií balíčkov, čo má často krát za následok skreslené výsledky skenu napríklad pre backportované balíčky ktoré síce majú číslo zraniteľnej verzie, avšak

bezpečnostné záplaty už boli pre túto staršiu verziu spätne aplikované, čo v praxi znamená že skenom nájdená zraniteľnosť je len heurestická, ale reálne v praxi neexistuje.

Na druhej strane je potrebné uviesť, že jeho využitie v PoC ostalo na úrovni základného skenovania a manuálnej analýzy výsledkov. Neboli plne implementované všetky plánované integračné väzby, napríklad automatizované prepojenie na SIEM, SOAR alebo ticketing systémy, a nástroj tak v tejto etape nefunguje ako plne integrovaný zdroj pre riadenie zraniteľností.

Z tohto dôvodu je Greenbone v PoC potrebné chápať ako funkčný, ale úplne oddelený samostatne pracujúci komponent. Napriek tomu jeho nasadenie preukázalo praktickú využiteľnosť a potvrdilo vhodnosť jeho zaradenia do širšej architektúry SOC služby.

2.5 OpenCTI

OpenCTI je v rámci PoC riešenia INNOSOC využitý ako ako centrálne úložisko hrozieb, kde sa zhromažďujú indikátory kompromitácie (IoC), TTP techniky, kampane a aktéri. Platforma umožňuje tieto dáta analyzovať a následne distribuovať do ďalších bezpečnostných nástrojov bez nutnosti manuálneho zásahu.

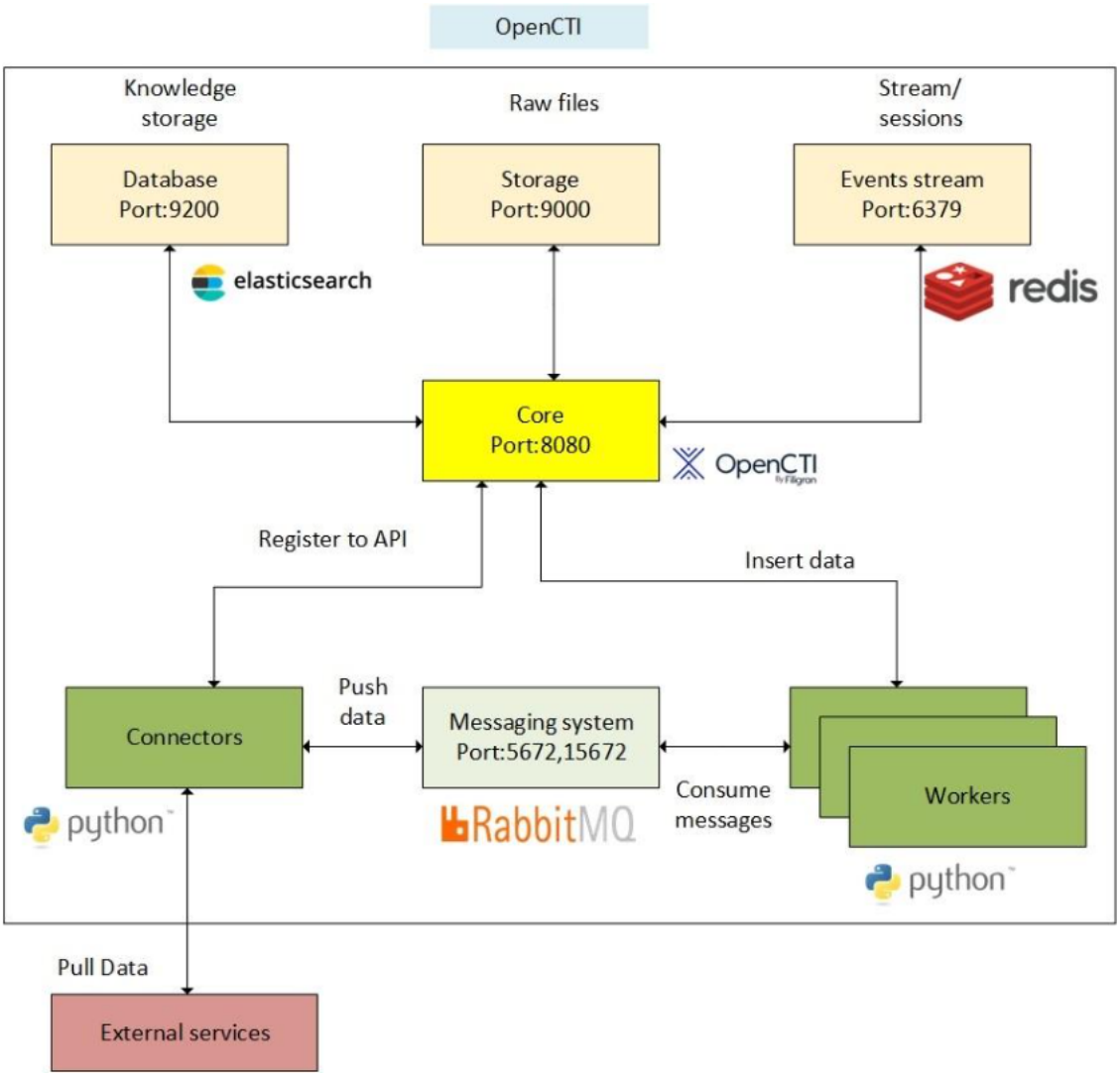
OpenCTI (Open Cyber Threat Intelligence Platform) je open-source platforma určená na správu, analýzu a zdieľanie informácií o kybernetických hrozbách (CTI). Umožňuje agregovať dáta z rôznych zdrojov, korelovať ich a transformovať do jednotného modelu založeného na štandarde STIX.

2.5.1 Nasadenie Shuffle v prostredí OpenStack

Technická špecifikácia nasadenia Shuffle:

- Prostredie – OpenStack
- Meno inštancie – OpenCTI_s
- IP adresa – 192.168.4.121
- vCPU – 4
- RAM – 16 GB
- Disk – 80 GB
- OS – Ubuntu 22.04

Schéma na obrázku znázorňuje architektúru platformy OpenCTI na serveri a tok spracovania dát medzi jednotlivými komponentmi. Centrálnym prvkom je OpenCTI Core bežiaci na porte 8080, ktorý zabezpečuje logiku aplikácie a komunikáciu medzi službami. Core komunikuje s databázou Elasticsearch na porte 9200, kde sa ukladajú všetky CTI dáta, ako aj s úložiskom súborov na porte 9000, ktoré slúži na ukladanie raw dát a príloh. Na správu eventov a cache sa využíva Redis na porte 6379.



V runtime prostredí sa nachádza messaging systém RabbitMQ (porty 5672 a 15672), ktorý zabezpečuje asynchrónnu komunikáciu medzi komponentmi. Konektory získavajú dáta z externých zdrojov, registrujú sa cez API do Core a následne posielajú dáta do RabbitMQ. Workers tieto správy spracovávajú, vykonávajú transformáciu a enrichment dát a následne ich ukladajú späť do OpenCTI cez Core.

2.5.2 Základné integračné väzby pre OpeCTI komponent

Aktuálne prepojené a otestované služby pomocou OpenCTI:

- **Abuse.ch (ThreatFox, URLhaus, Malwarebazzar)** – automatický import IoC
- **Ransomwarelive** – import strategických CTI
- **MITRE ATT&CK** – mapovanie techník a taktík
- **SOAR (Shuffle)** – automatizácia reakcií na incidenty

2.5.3 Poznámka k platforme MISP

V rámci pôvodného architektonického návrhu bola ako samostatný komponent pre zber a zdieľanie informácií o hrozbách (Threat Intelligence) zvažovaná aj platforma MISP (Malware Information Sharing Platform). Po vyhodnotení aktuálnych možností a rozsahu PoC bolo rozhodnuté, že zber, správa a distribúcia CTI dát bude v rámci PoC riešená výhradne prostredníctvom platformy OpenCTI.

OpenCTI natívne podporuje integráciu s MISP ako jedným z externých zdrojov dát prostredníctvom MISP konektora, ktorý umožňuje automatický import indikátorov kompromitácie (IoC) a ďalších štruktúrovaných dát z MISP inštancií. Táto integrácia však v rámci PoC nebola aktivovaná ani testovaná, keďže pre overenie konceptu postačuje priame napojenie OpenCTI na vybrané verejné zdroje hrozieb (Abuse.ch, MITRE ATT&CK, Ransomwarelive).

Napojenie MISP ako externej databázy hrozieb zostáva možnosťou pre budúce rozšírenie riešenia v produkčnom prostredí, kde by mohlo poskytnúť prístup k širšej komunite zdieľania CTI dát a k interným databázam organizácie.

2.6 GLPI

2.6.1 Úloha GLPI v PoC

GLPI je v rámci PoC riešenia INNOSOC využitý ako centrálny nástroj pre evidenciu IT aktív, správu konfigurácií (CMDB) a riadenie IT služieb (ITSM). V navrhovanom SOC riešení neplní úlohu detekčného ani reakčného komponentu, ale predstavuje procesnú a evidenčnú vrstvu, ktorá prepája technické bezpečnostné udalosti s organizačným kontextom, vlastníkmi aktív a poskytovanými službami. Jeho význam spočíva najmä v tom, že umožňuje systematicky evidovať aktíva, ich vzťahy, zodpovednosti a zároveň zabezpečiť riadenie incidentov, požiadaviek a zmien prostredníctvom jednotného ticketing systému.

V rámci PoC je GLPI chápané ako nástroj, ktorý dopĺňa technické údaje získané z detekčných a analytických komponentov (napr. SIEM, EDR) o procesný a organizačný kontext. Pri spracovaní bezpečnostných udalostí tak umožňuje priradiť incident ku konkrétnemu aktívu, identifikovať jeho vlastníka, určiť zodpovednosti a sledovať celý životný cyklus riešenia od nahlásenia až po uzavretie. Takýto prístup zvyšuje transparentnosť, auditovateľnosť a konzistentnosť riešenia bezpečnostných incidentov.

GLPI v SOC poskytuje procesný rámec, v ktorom sú bezpečnostné udalosti formalizované a riadené. V tomto zmysle predstavuje doplnkový, no kľúčový komponent riešenia, ktorý zabezpečuje prepojenie technickej a organizačnej roviny a vytvára predpoklad pre efektívne riadenie bezpečnostných procesov a ďalšie integračné rozšírenia.

2.6.2 Nasadenie a základná konfigurácia GLPI

V rámci PoC riešenia INNOSOC bol nástroj GLPI nasadený ako samostatná webová aplikácia určená pre evidenciu IT aktív a riadenie servisných procesov. Z pohľadu implementácie bol systém prevádzkovaný nad štandardným webovým stackom (Apache/Nginx, PHP) s využitím relačnej databázy (MySQL/MariaDB), čo zodpovedá jeho

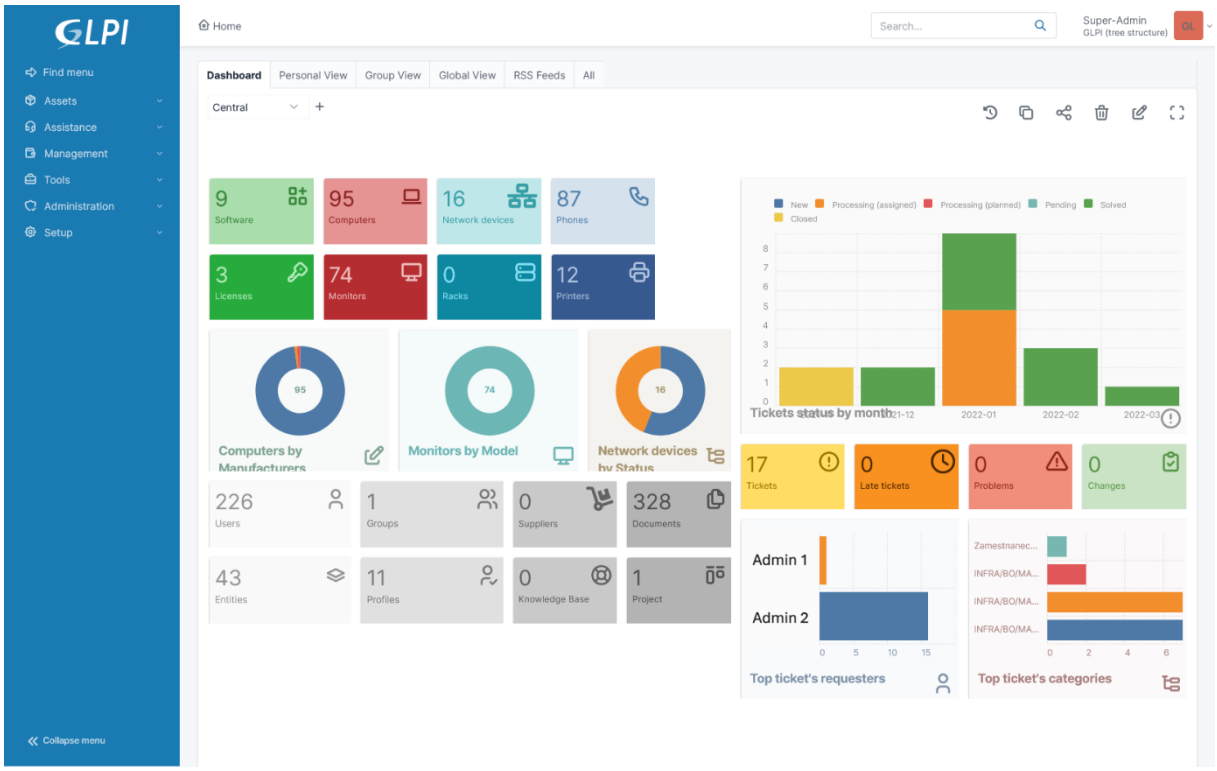
viacvrstvovej architektúre a umožňuje jednoduché nasadenie aj škálovanie. Takýto spôsob implementácie je vhodný pre testovacie aj produkčné prostredie a zároveň umožňuje integráciu s ďalšími komponentmi SOC.

Základná konfigurácia GLPI bola v PoC zameraná na sprístupnenie webového rozhrania a aktiváciu kľúčových funkcionalít, najmä evidencie aktív (CMDB) a ticketing systému. Systém bol konfiguračne pripravený na správu používateľov, rolí a väzieb medzi aktívami, pričom databázová vrstva zabezpečuje ukladanie prevádzkových a historických údajov. V rámci PoC boli do systému zavedené základné objekty, ako IT aktíva, používatelia, organizačné jednotky a incidentné záznamy, ktoré umožňujú prepojenie technických udalostí s procesným a organizačným kontextom.

2.6.3 Využitie GLPI v rámci PoC

V rámci PoC bol GLPI využívaný ako procesná a evidenčná vrstva pre správu incidentov a IT aktív v kontexte SOC riešenia. Jeho hlavný prínos spočíval v tom, že umožňoval priradiť technické bezpečnostné udalosti ku konkrétnym aktívam, používateľom a organizačným jednotkám a zároveň zabezpečiť ich formálne spracovanie prostredníctvom ticketing systému.

Praktické využitie GLPI v tejto etape bolo orientované najmä na evidenciu aktív, správu základných vzťahov v CMDB a sledovanie životného cyklu incidentov. GLPI tak neplnil úlohu detekčného ani reakčného nástroja, ale dopĺňal riešenie o procesný a organizačný kontext potrebný na riadenie incidentov. V tomto rozsahu jeho nasadenie potvrdilo, že ide o vhodný doplnok PoC riešenia a o použiteľný základ pre ďalšie rozšírenie integračných a automatizačných scenárov.

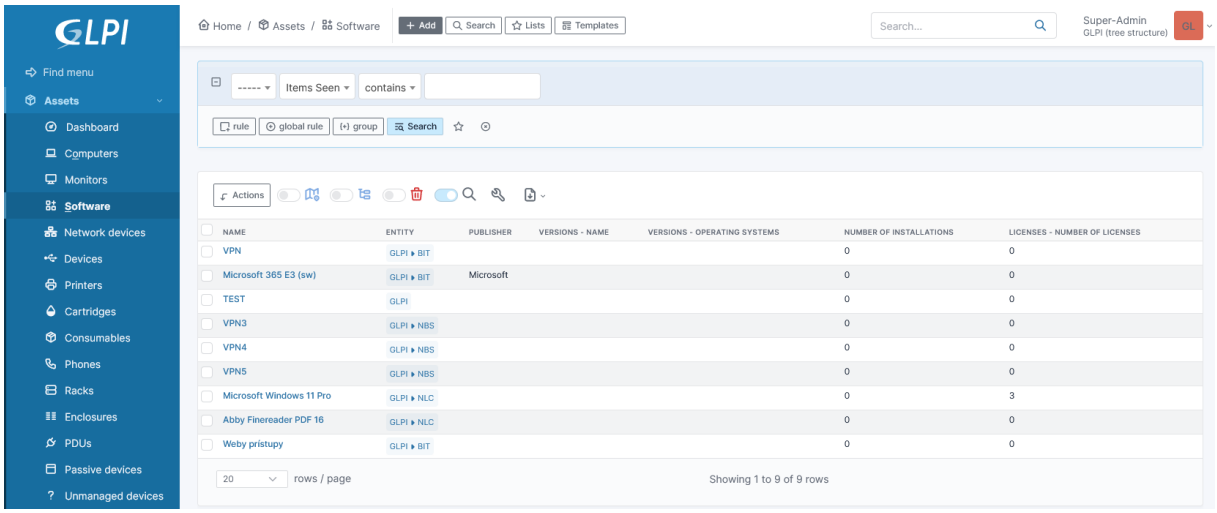


Obrázok zobrazuje dashboard nástroja GLPI, ktorý poskytuje prehľad o evidovaných IT aktívach a stave ITSM procesov.

V hornej časti sú sumarizované kategórie aktív (napr. počítače, sieťové zariadenia, softvér), čím je viditeľná úloha GLPI ako CMDB systému. Stredná časť obsahuje štatistiky a prehľady infraštruktúry, doplnené o informácie o používateľoch a organizačných entitách.

Na pravej strane sa nachádza sekcia ticketingu, ktorá zobrazuje stav incidentov a požiadaviek (nové, spracovávané, uzavreté) vrátane základných metrík.

Z pohľadu SOC architektúry GLPI predstavuje procesnú a evidenčnú vrstvu, ktorá prepája technické udalosti s aktívami a zabezpečuje riadené a auditovateľné spracovanie incidentov.



Obrázok zobrazuje rozhranie nástroja GLPI v sekcii správy softvéru (Assets → Software), ktorá slúži na evidenciu a správu nainštalovaných aplikácií v prostredí organizácie.

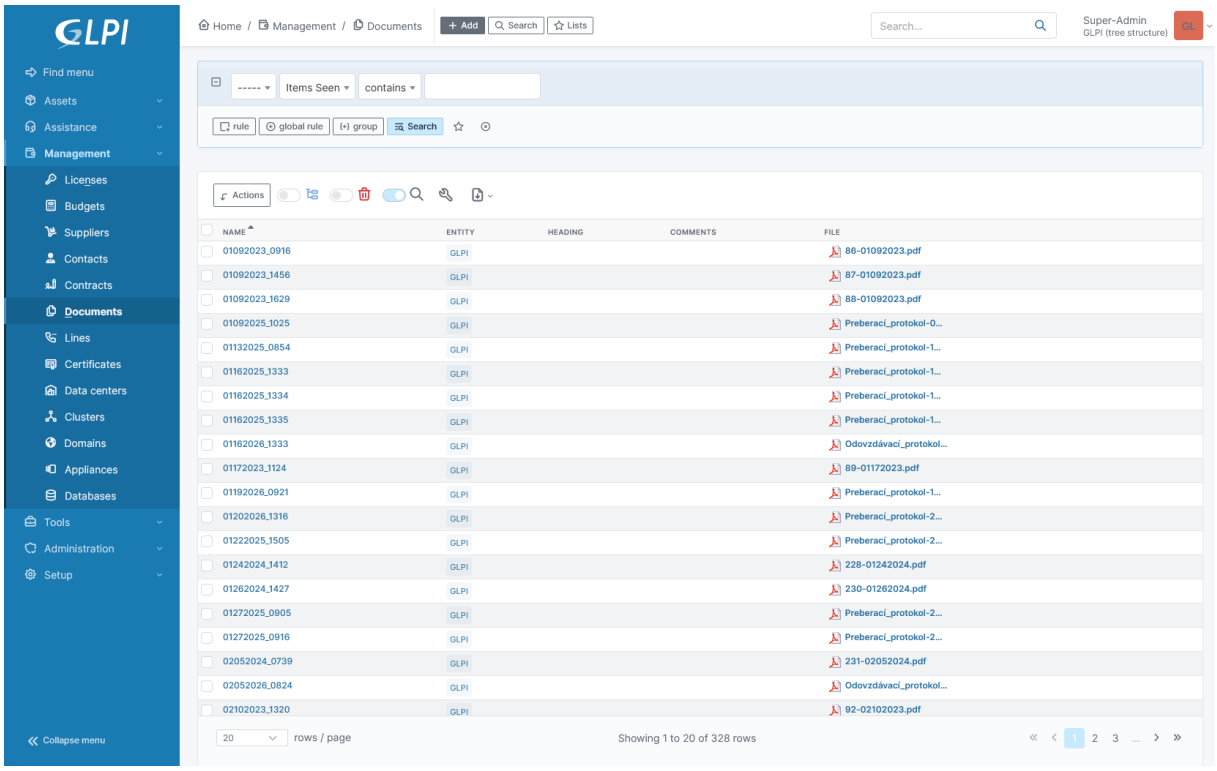
V centrálnej časti sa nachádza tabuľkový prehľad softvérových položiek (napr. VPN, Microsoft 365, Windows 11), pričom každý záznam obsahuje:

- názov softvéru,
- organizačnú entitu (Entity),
- vydavateľa (Publisher),
- verzie a podporované operačné systémy,
- počet inštalácií,
- informácie o licenciách.

Horná časť rozhrania umožňuje filtrovanie a vyhľadávanie záznamov, zatiaľ čo bočný panel poskytuje navigáciu medzi rôznymi typmi aktív (počítače, siete, zariadenia a pod.).

Z pohľadu SOC architektúry táto sekcia reprezentuje evidenciu softvérových aktív a licencií, ktorá je kľúčová pre:

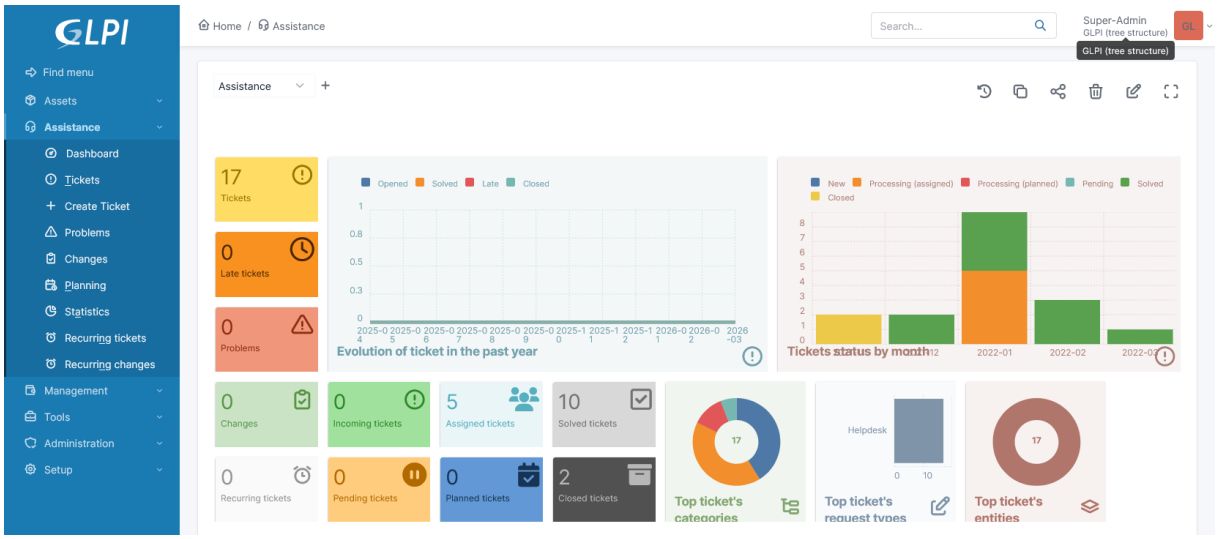
- identifikáciu zraniteľného alebo zastaraného softvéru,
- väzbu na konkrétne zariadenia a vlastníkov,
- podporu riadenia zraniteľností a incidentov v kontexte konkrétnych aplikácií.



Obrázok zobrazuje sekciu **Documents v nástroji GLPI**, ktorá slúži na správu a evidenciu dokumentácie v rámci IT prostredia.

V centrálnej časti sa nachádza tabuľkový prehľad dokumentov s identifikátormi, priradenou entitou a odkazmi na súbory (najčastejšie PDF). Rozhranie umožňuje filtrovanie, vyhľadávanie a základné operácie nad dokumentmi.

Táto funkcionalita zabezpečuje **centralizované ukladanie a dohľadateľnosť dokumentov**, ako sú protokoly, záznamy alebo zmluvná dokumentácia, čo je dôležité z pohľadu auditu a procesného riadenia v SOC prostredí.



Obrázok zobrazuje **dashboard modulu Assistance v nástroji GLPI**, ktorý slúži na operatívne riadenie a monitorovanie ticketov (incidentov a požiadaviek) v rámci ITSM.

V ľavej časti sa nachádza navigačné menu so sekciami ako Dashboard, Tickets, Problems, Changes či Planning, pričom aktívna je oblasť Assistance. Centrálna časť dashboardu poskytuje súhrnné metriky ticketov prostredníctvom farebne odlíšených panelov – napríklad celkový počet ticketov (17), počet oneskorených ticketov, problémov a zmien. Tieto metriky poskytujú rýchly prehľad o aktuálnom stave servisných požiadaviek.

Dashboard ďalej obsahuje vizualizačné prvky:

- časový graf zobrazujúci vývoj ticketov za sledované obdobie,
- stĺpcový graf reprezentujúci stav ticketov podľa mesiacov (nové, spracovávané, vyriešené a pod.),
- doplnkové štatistiky ako počet priradených, vyriešených či čakajúcich ticketov.

Spodná časť obsahuje agregované pohľady na kategórie ticketov, typy požiadaviek a organizačné entity, čo umožňuje identifikovať najviac zaťažené oblasti alebo najčastejšie typy incidentov.

Z pohľadu SOC architektúry tento dashboard reprezentuje **procesnú a riadiacu vrstvu incident managementu**, kde sú bezpečnostné udalosti transformované na ticketové záznamy, sledovaný ich životný cyklus a vyhodnocovaná efektivita riešenia.

2.6.4 Aktuálny stav a obmedzenia GLPI v SoC

GLPI bol v rámci PoC úspešne nasadený a základne nakonfigurovaný ako funkčný nástroj pre evidenciu IT aktív a riadenie incidentov. V tomto stave poskytuje použiteľný základ pre správu CMDB údajov, evidenciu vzťahov medzi aktívami a podporu procesov ticketingu. Z pohľadu PoC sa potvrdilo, že GLPI je vhodný na doplnenie SOC riešenia o procesný a organizačný kontext a na podporu riadenia bezpečnostných incidentov.

Na druhej strane je potrebné uviesť, že jeho využitie v PoC ostalo na úrovni základnej evidencie a spracovania ticketov. Neboli plne implementované všetky plánované integračné väzby voči ostatným komponentom riešenia (napr. automatizované prepojenie na SIEM/SOAR), a GLPI tak v tejto etape nefunguje ako plne integrovaný nástroj pre riadenie

incidentov s automatickým obohacovaním dát. Jeho úloha je zatiaľ skôr podporná a pripravuje základ pre ďalšie rozšírenie v budúcich etapách riešenia.

Z tohto dôvodu je GLPI v PoC potrebné chápať ako funkčný, ale zatiaľ nie plne rozvinutý komponent. Napriek tomu jeho nasadenie preukázalo praktickú využiteľnosť v navrhovanom riešení a potvrdilo vhodnosť jeho zaradenia do širšej architektúry SOC služby.

2.7 NetBox

2.7.1 Úloha NetBox v PoC

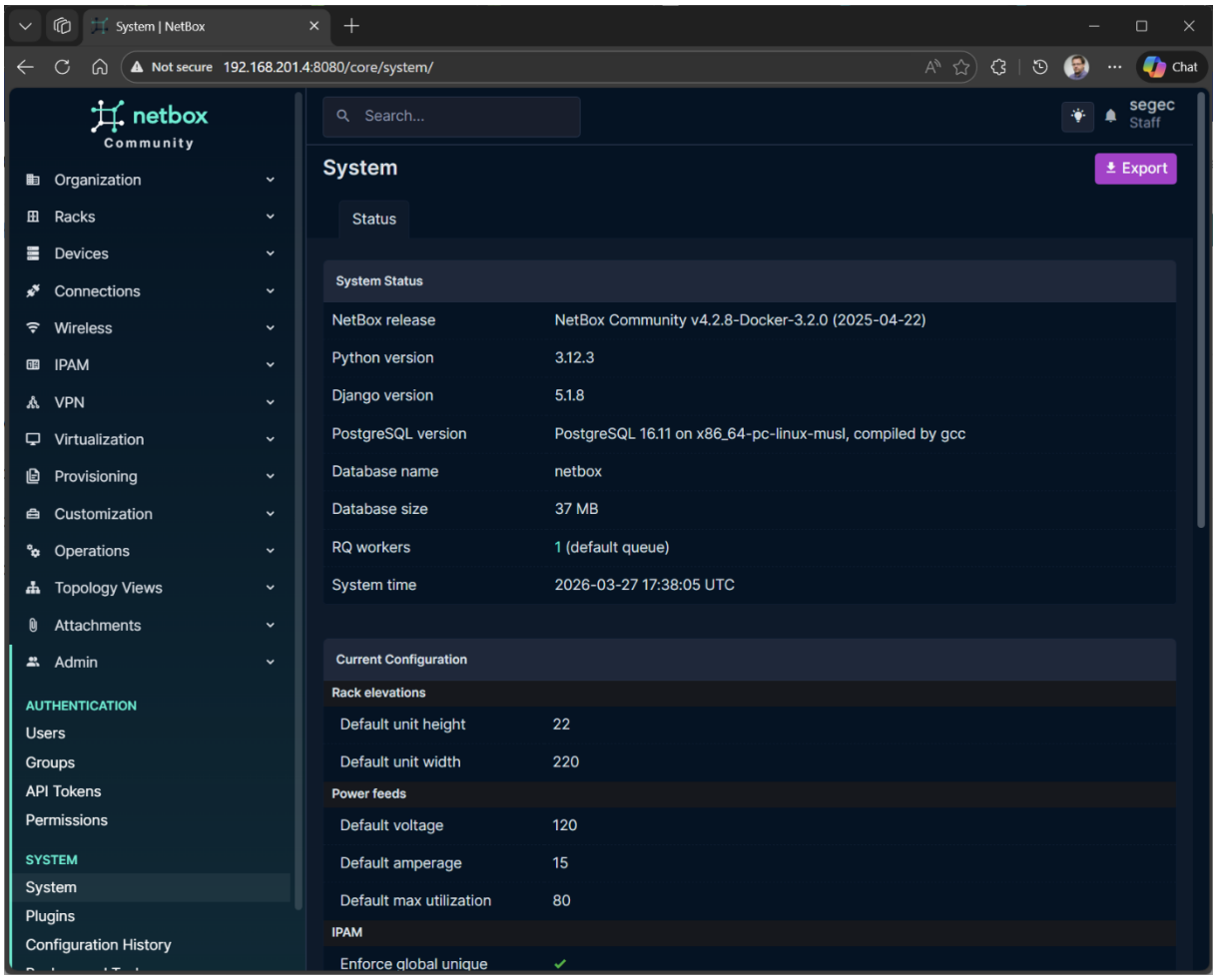
NetBox je v rámci PoC riešenia INNOSOC využitý ako podporný nástroj pre evidenciu a správu infraštruktúrneho kontextu. V navrhovanom SOC riešení neplní úlohu detekčného ani reakčného komponentu, ale slúži ako zdroj informácií o aktívach, adresácii, sieťových segmentoch a vybraných väzbách medzi prvkami infraštruktúry. Jeho význam spočíva najmä v tom, že umožňuje lepšie pochopiť, ku ktorému zariadeniu, systému alebo časti siete sa konkrétna udalosť vzťahuje. V rámci PoC je NetBox chápaný ako kontextová vrstva, ktorá dopĺňa technické údaje získané z ostatných komponentov riešenia. Pri analýze bezpečnostných udalostí tak môže slúžiť na dohľadanie základných informácií o zariadeniach, IP rozsahoch, lokalitách alebo logickom zaradení prvkov infraštruktúry. Takýto prístup zvyšuje prehľadnosť riešenia a vytvára predpoklad na presnejšie vyhodnocovanie bezpečnostných situácií.

NetBox v SOC ponúka kontext prostredia, v ktorom bezpečnostné logy a alerty vznikajú. V tomto zmysle predstavuje ***doplnkový***, ale dôležitý komponent riešenia, ktorý podporuje orientáciu v infraštruktúre a pripravuje základ pre ďalšie integračné využitie v rozšírených scenároch.

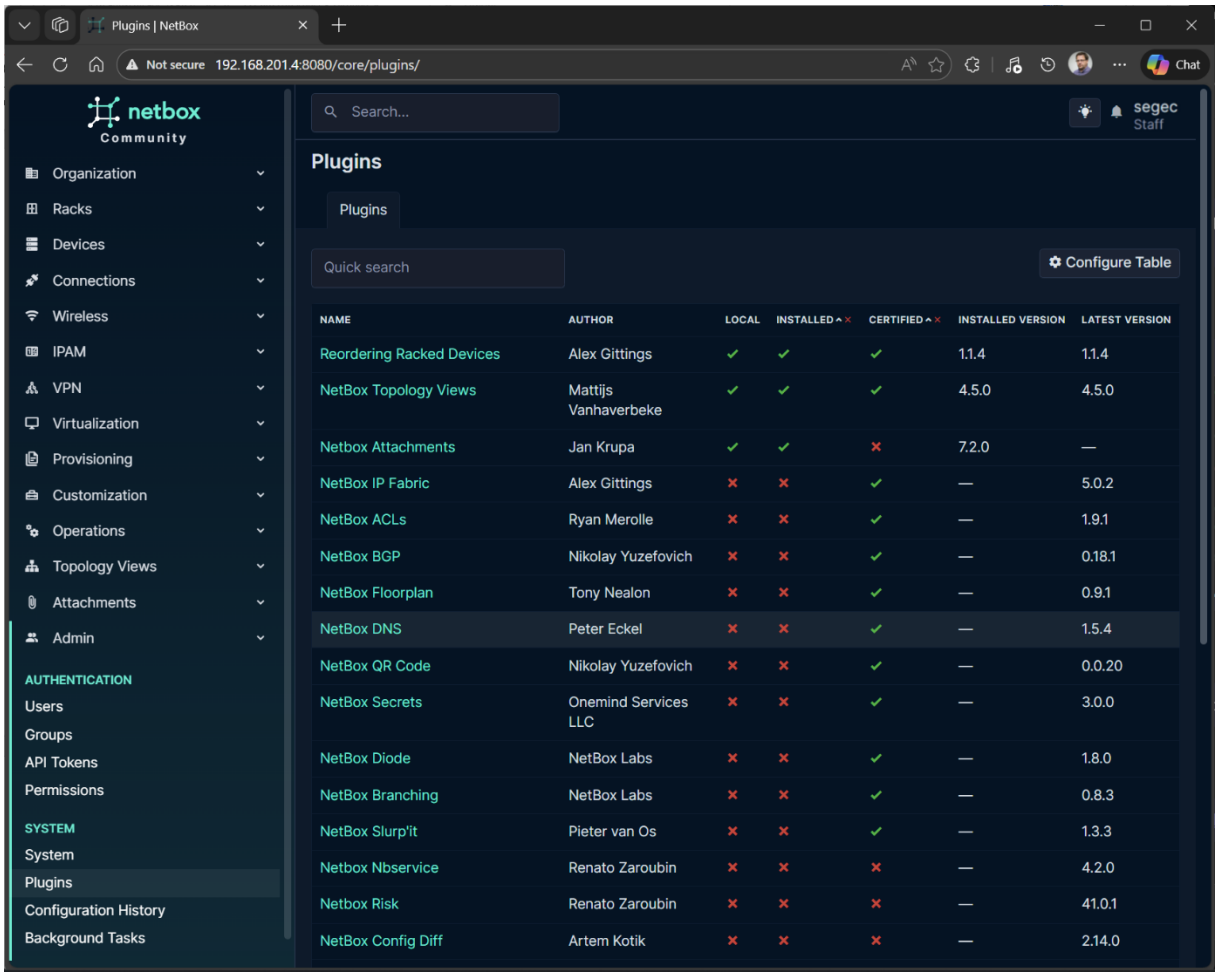
2.7.2 Nasadenie a základná konfigurácia NetBoxu

V rámci PoC bol NetBox nasadený v prostredí OpenStack ako samostatná služba určená pre evidenciu a správu infraštruktúrnych údajov. Z pohľadu implementácie bol riešený kontajnerizačne, pričom boli využité aplikačné a podporné komponenty potrebné na jeho beh, najmä databázová vrstva a cache/queue služby. Takýto spôsob nasadenia zodpovedá charakteru experimentálneho nasadenia a zároveň umožňuje jednoduchšiu správu, aktualizáciu a opakovateľné nasadenie.

Základná konfigurácia NetBoxu bola v PoC zameraná na sprístupnenie webového rozhrania a na prípravu dátového modelu pre evidenciu vybraných objektov infraštruktúry. Konfiguračne bol systém pripravený na prácu s databázou PostgreSQL a s oddelenou Redis vrstvou pre úlohy a cache. V rámci PoC boli do systému zavedené základné objekty, najmä lokality, zariadenia, adresné priestory a súvisiaci infraštruktúrny kontext.



Obrázok 1 Informácie o systéme



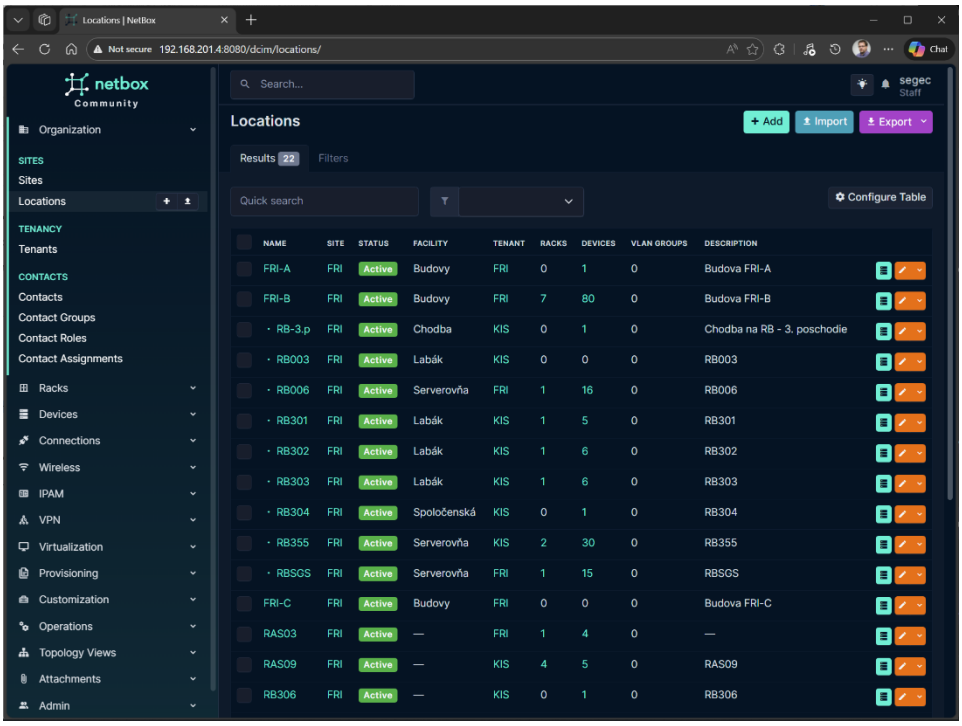
Obrázok 2 Pluginy

Cieľom nebolo vytvoriť úplnú produkčnú CMDB alebo detailný model celej siete, ale pripraviť reprezentatívnu množinu údajov, na ktorej je možné demonštrovať využitie NetBoxu v navrhovanom riešení.

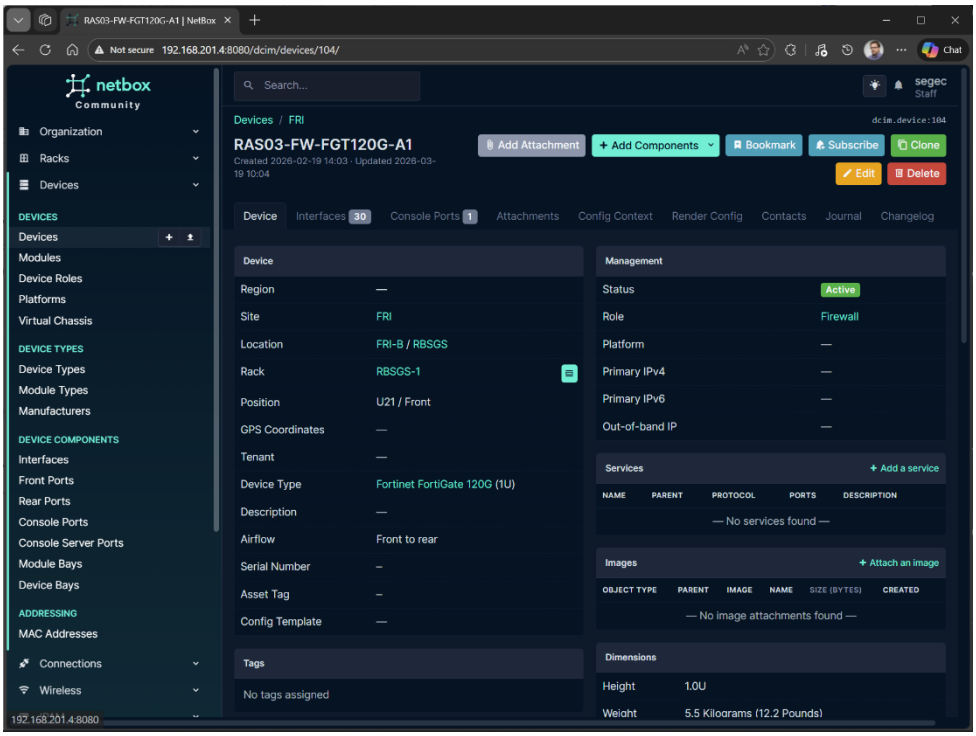
2.7.3 Využitie NetBoxu v rámci PoC

V rámci PoC bol NetBox využívaný ako podporný zdroj infraštruktúrneho kontextu pre ostatné časti riešenia. Jeho hlavný prínos spočíval v tom, že umožňoval dohľadanie základných informácií o zariadeniach, IP adresácii, lokalitách a sieťových segmentoch, ku ktorým bolo možné priradiť udalosti alebo zistenia z ďalších nástrojov.

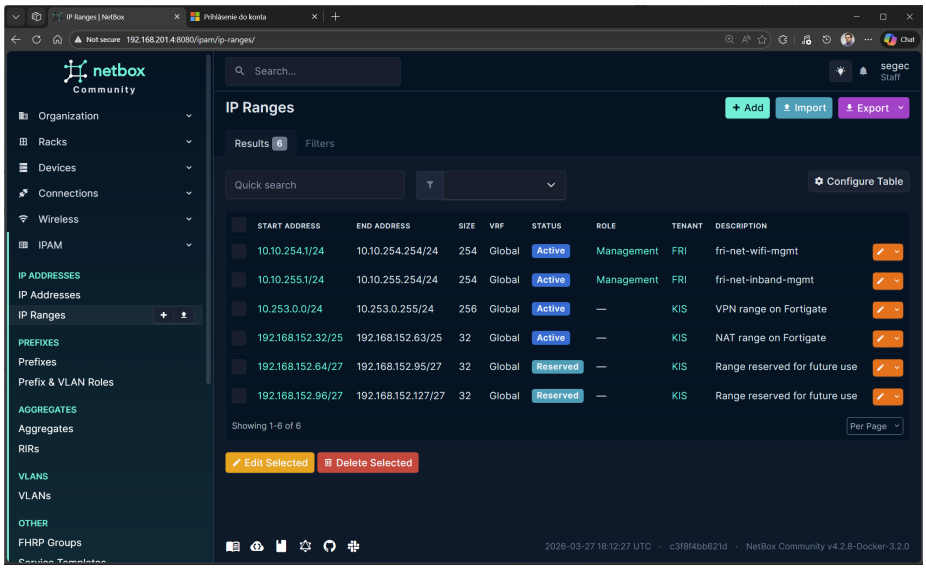
Praktické využitie NetBoxu v tejto etape bolo orientované najmä na evidenciu a manuálne alebo čiastočne podporené dohľadanie kontextu. NetBox tak neplnil úlohu detekčného ani reakčného nástroja, ale dopĺňal riešenie o údaje potrebné na lepšie pochopenie technického prostredia. V tomto rozsahu jeho nasadenie potvrdilo, že ide o vhodný doplnok PoC riešenia a o použiteľný základ pre ďalšie rozšírenie integračných scenárov.



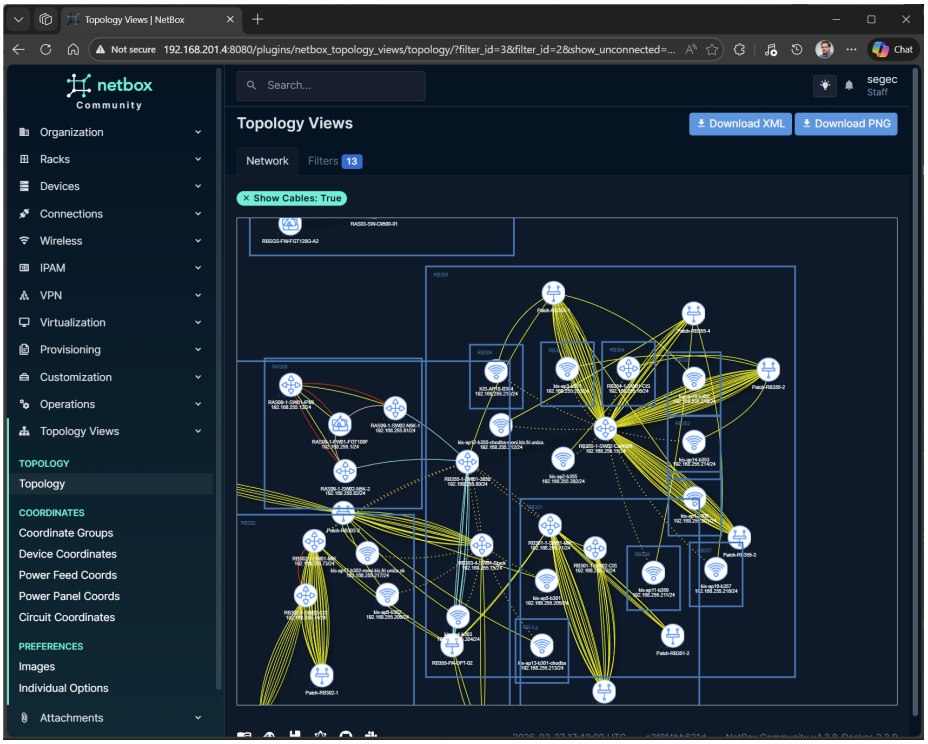
Obrázok 3 Modelovanie organizácie

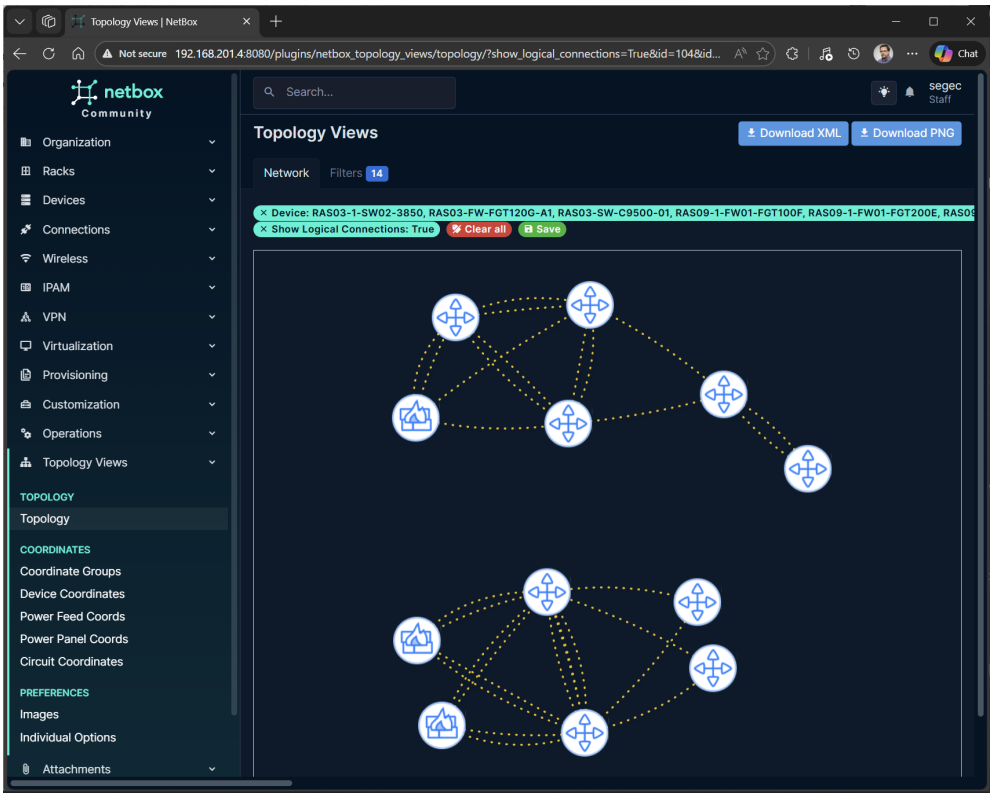


Obrázok 4 Modelovanie zariadenia – Firewallu

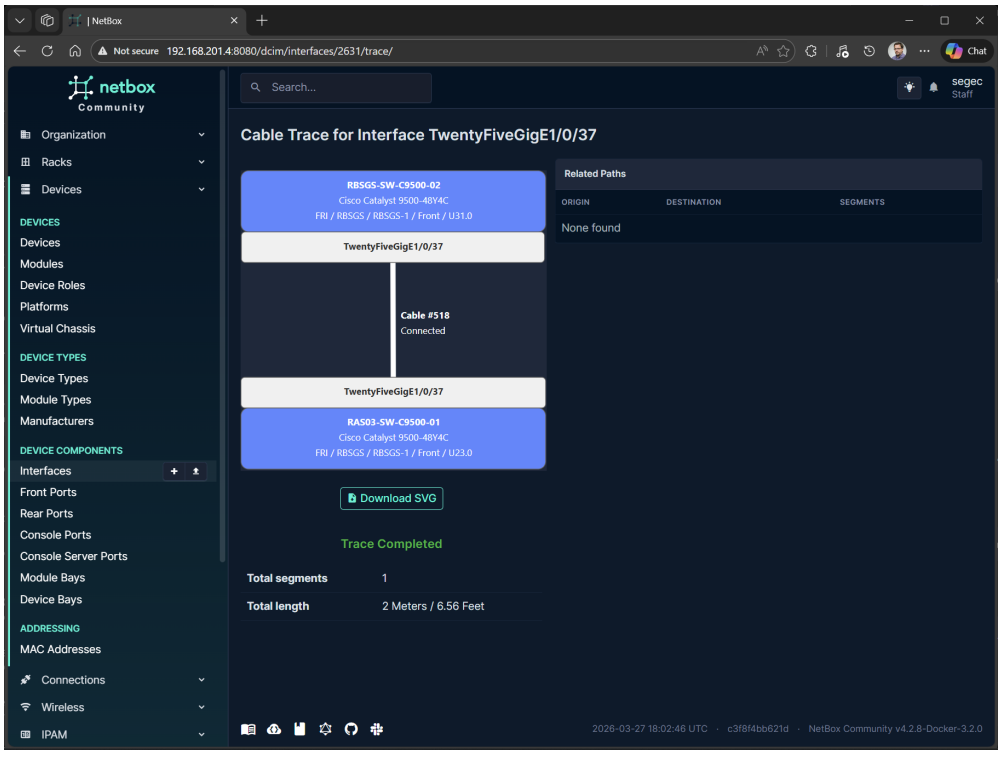


Obrázok 5 Manažment adries





Obrázok 6 Generovanie topológií



Obrázok 7 Trasovanie kabeláže

2.7.4 Aktuálny stav a obmedzenia NetBoxu v PoC

NetBox bol v rámci PoC úspešne nasadený a základne nakonfigurovaný ako funkčný nástroj pre evidenciu infraštruktúrnych údajov. V tomto stave poskytuje použiteľný základ pre správu vybraných objektov a pre orientáciu v prostredí navrhovaného riešenia. Z pohľadu PoC sa potvrdilo, že NetBox je vhodný na doplnenie SOC riešenia o infraštruktúrny kontext a na podporu práce s aktívami a adresáciou.

Na druhej strane je potrebné uviesť, že jeho využitie v PoC ostalo na úrovni základnej evidencie a čiastočného kontextového doplnenia. Neboli dotiahnuté všetky plánované integračné väzby voči ostatným komponentom riešenia a NetBox tak v tejto etape nefunguje ako plne automatizovaný zdroj enrichment údajov. Jeho úloha je zatiaľ skôr podporná a pripravuje základ pre ďalšie rozšírenie v budúcich etapách riešenia.

Z tohto dôvodu je NetBox v PoC potrebné chápať ako funkčný, ale zatiaľ nie plne rozvinutý komponent. Napriek tomu jeho nasadenie preukázalo praktickú využiteľnosť v navrhovanom riešení a potvrdilo vhodnosť jeho zaradenia do širšej architektúry SOC služby.

2.8 DFIR IRIS

DFIR-IRIS je v rámci PoC riešenia INNOSOC využitý ako open-source nástroj určený na riadenie a evidenciu bezpečnostných incidentov (case management). V rámci nášho PoC slúži ako centrálne miesto, kde sú prijímané alerty zo systému Wazuh a následne spracovávané ako incidenty (cases). Umožňuje evidenciu artefaktov, poznámok, úloh, zmenu stavov incidentov a koordináciu práce analytikov.

2.8.1 Model nasadenia DFIR IRIS v prostredí OpenStack

Systém je nasadený v prostredí OpenStack na virtuálnom stroji s operačným systémom Ubuntu 24.04. DFIR-IRIS beží v kontajneroch pomocou nástroja Docker Compose, čo umožňuje jednoduché nasadenie, správu aj škálovanie riešenia.

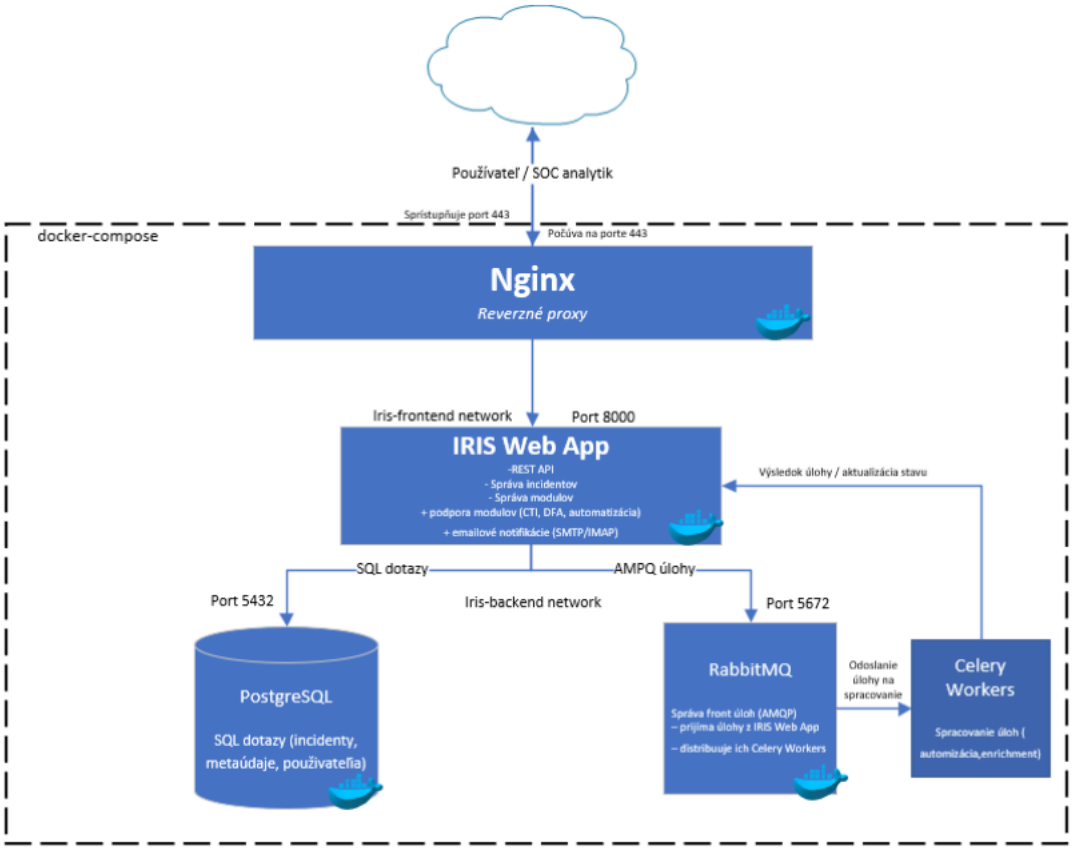
Technická špecifikácia nasadenia DFIR-IRIS:

- Prostredie – OpenStack
- Meno inštancie – ubuntu-24.04-kis
- IP adresa – 192.168.20.45 (Interná IP v Openstacku), 192.168.4.97 (Externá IP)
- vCPU – 4
- RAM – 8 GB
- Disk – 40 GB
- OS – Ubuntu 24.04

Riešenie je postavené na kontajnerovej architektúre, kde jednotlivé komponenty (webové rozhranie, backend a databázová vrstva) bežia oddelene v Docker kontajneroch. Spustenie celej aplikácie je realizované pomocou príkazu:

docker compose up -d

DFIR-IRIS poskytuje webové rozhranie, cez ktoré analytik pracuje s incidentmi, pričom backend zabezpečuje spracovanie logiky aplikácie a komunikáciu s databázou, kde sú ukladané všetky incidenty, artefakty a história zmien



Obrázok 2 Architektúra DFIR-IRIS

Obrázok znázorňuje architektúru nástroja DFIR-IRIS a spracovanie incidentov v rámci INNOSOC prostredia. Riešenie je založené na kontajnerizovanej architektúre, kde jednotlivé komponenty komunikujú prostredníctvom interných služieb.

Frontend predstavuje webové rozhranie pre analytikov, ktoré slúži na prácu s incidentmi (cases), artefaktami a úlohami. Požiadavky z rozhrania sú spracované backendom, ktorý zabezpečuje aplikačnú logiku a riadenie workflow incidentov. Backend komunikuje s databázovou vrstvou, kde sú uložené všetky informácie o incidentoch vrátane stavu, používateľov, artefaktov a časovej osi. Táto architektúra umožňuje konzistentnú evidenciu a spätné dohľadanie vykonaných operácií.

2.8.2 Základné integračné väzby komponentu DFIR IRIS v INNOSOC

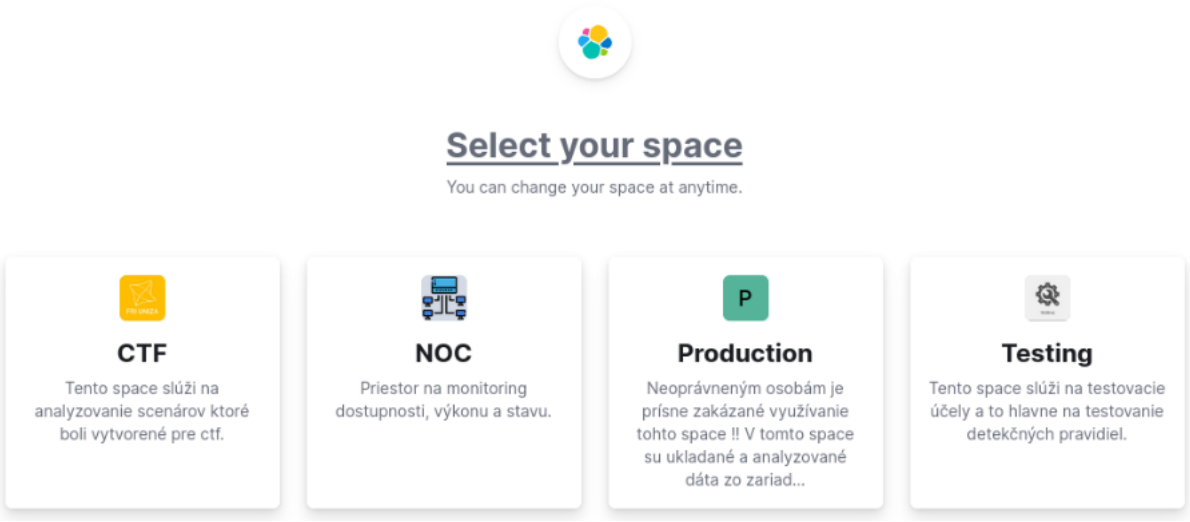
DFIR-IRIS podporuje integráciu pomocou REST API a webhookov. V rámci PoC je prepojený so systémom Wazuh, z ktorého prijíma alerty vo formáte JSON a spracováva ich ako incidenty alebo aktualizácie existujúcich prípadov. Systém Shuffle je s DFIR-IRIS prepojený a bude slúžiť ako automatizačná vrstva pre operácie nad incidentmi, ako napríklad enrichment, aktualizácia prípadov alebo spúšťanie reakcií. Toto prepojenie je implementované a jeho funkčnosť bola overovaná testovaním.

Aktuálne prepojené a otestované služby pomocou DFIR-IRIS:

- Wazuh – príjem alertov a vytváranie incidentov (JSON / webhook)
- Shuffle – automatizácia a aktualizácia incidentov pomocou API

2.9 Kibana (ELK)

Kibana je v rámci PoC riešenia INNOSOC využitá ako grafické rozhranie v ELK stack. Na FRI UNIZA je nasadené na url adrese v doméne FRI. Logické delenie pravidiel, prístupov a iné, je realizované pomocou Spaces. Nasadené máme nasledovné Spaces. Na obrázku je vidieť aj ich úlohu / využitie.



Hlavné funkcie Kibany využité v nasadenom komponente:

- Discover – prezeranie záznamov,
- Dashboards (Analytics) – zobrazovanie záznamov do prehľadných grafov,
- Dashboards (Security) - fungovanie pravidiel,
- Rules - správa detekčných pravidiel,
- Alerts,
- Dev Tools - konzola na správu,
- Stack Monitoring,
- Stack Management.

V Stack Managment sa spravuje celý stack.

2.9.1 Vytvorené dashboardy

Dashboardy v Kibane slúžia na vizualizáciu dát z Elasticsearch pomocou grafov, máp a filtrov, čím poskytujú prehľad o sledovaných údajoch.

Vytvorené dashboardy:

- Suricata UNIZA
- Arkime UNIZA
- F_kibana_dashboards
- Noc výskumníkov
- IDS DMZ: Vizualizuje dáta z IDS senzorov v demilitarizovaných zónach a umožňuje porovnávať detekcie medzi nástrojmi Suricata a Fortigate.
- Login Activity Monitoring: Zameriava sa na úspešné a neúspešné prihlásenia
- Zaťaženie Stacku: Slúži na monitorovanie využitia operačnej pamäte jednotlivými uzlami Elasticsearch, Logstash a Kibany



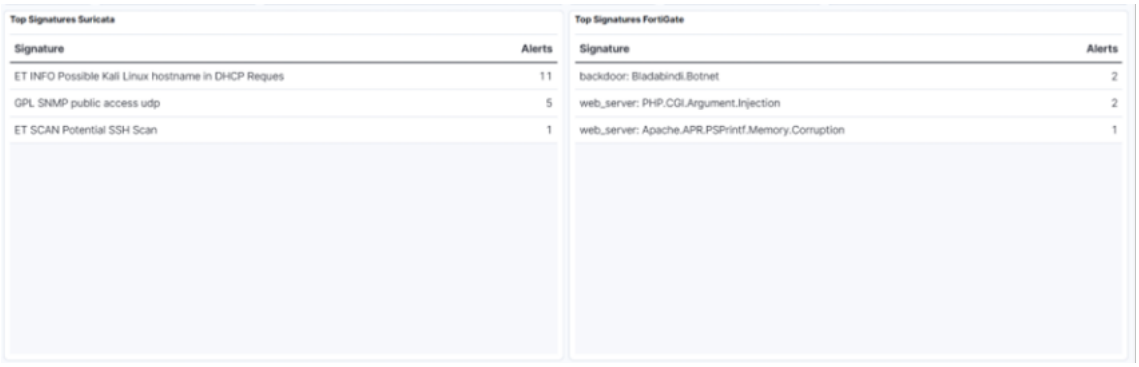
Dashboard IDS DMZ, alerty

Obrázok zobrazuje dashboard v nástroji Kibana, ktorý slúži na vizualizáciu bezpečnostných udalostí detegovaných IDS systémami, konkrétne Suricata a FortiGate.

Ľavá časť dashboardu prezentuje dáta zo systému Suricata IDS, kde je zobrazený celkový počet alertov (17), ich rozdelenie podľa závažnosti a časový priebeh detekcií za sledované obdobie. Grafické prvky umožňujú rýchlu identifikáciu nárastu alebo poklesu bezpečnostných udalostí.

Pravá časť dashboardu zobrazuje obdobné metriky pre FortiGate IDS, vrátane celkového počtu alertov (5), ich klasifikácie podľa závažnosti a vývoja v čase.

Z pohľadu SOC architektúry tento dashboard predstavuje vizualizačnú a analytickú vrstvu, ktorá agreguje výstupy z detekčných nástrojov a umožňuje ich prehľadné vyhodnocovanie. Kibana tu neplní detekčnú funkciu, ale poskytuje analytikom nástroj na interpretáciu udalostí, sledovanie trendov a podporu rozhodovania pri riešení bezpečnostných incidentov.



Dashboard IDS DMZ, top signatúry

Obrázok zobrazuje dashboard v nástroji Kibana zameraný na prehľad najčastejšie detegovaných signatúr (Top Signatures) zo systémov Suricata a FortiGate IDS.

Ľavá časť obsahuje zoznam najčastejších signatúr zo Suricata IDS spolu s počtom zachytených alertov. Viditeľné sú napríklad udalosti súvisiace s DHCP komunikáciou, SNMP prístupom alebo pokusmi o SSH skenovanie. Tento prehľad umožňuje identifikovať najčastejšie typy detekovaných aktivít v sieti.

Pravá časť zobrazuje obdobný prehľad pre FortiGate IDS, kde sú uvedené signatúry súvisiace napríklad s botnet aktivitou alebo útokmi na webové aplikácie (napr. PHP injection či zraniteľnosti Apache servera).

Z pohľadu SOC architektúry tento dashboard predstavuje analytickú vrstvu, ktorá umožňuje rýchlu identifikáciu najvýznamnejších hrozieb a trendov v detekciách. Pomáha analytikom určiť, ktoré typy útokov alebo anomálií sa v prostredí vyskytujú najčastejšie, a tým podporuje prioritizáciu ďalšej analýzy a reakcie.



Dashboard IDS DMZ, top source IPs

Obrázok zobrazuje dashboard v nástroji Kibana zameraný na analýzu zdrojových IP adries (Top Source IPs) detegovaných IDS systémami Suricata a FortiGate.

Ľavá časť prezentuje dáta zo Suricata IDS, kde sú vizualizované najčastejšie zdrojové IP adresy generujúce bezpečnostné udalosti spolu s ich výskytom v čase. Graf umožňuje identifikovať dominantné zdroje komunikácie alebo potenciálne podozrivé aktivity.

Pravá časť zobrazuje obdobný prehľad pre FortiGate IDS, pričom jednotlivé IP adresy sú farebne odlíšené a zobrazené v časovej osi. To umožňuje sledovať, kedy a ako často jednotlivé zdroje generovali alerty.

Z pohľadu SOC architektúry tento dashboard predstavuje analytický nástroj na identifikáciu najaktívnejších alebo potenciálne rizikových zdrojov komunikácie. Umožňuje analytikom rýchlo odhaliť podozrivé IP adresy, sledovať ich správanie v čase a podporuje proces vyšetrovania bezpečnostných incidentov.



Dashboard IDS DMZ, mapa

Obrázok zobrazuje dashboard v nástroji Kibana vo forme geografickej mapy, ktorá vizualizuje bezpečnostné udalosti detegované IDS systémom (napr. Suricata) na základe geografického pôvodu IP adries.

Mapa pokrýva oblasť Európy a jednotlivé body reprezentujú lokality, z ktorých bola zaznamenaná podozrivá aktivita. Farebné odlíšenie bodov vyjadruje úroveň závažnosti (severity), čo umožňuje rýchlu orientáciu v rizikovosti jednotlivých udalostí.

V pravej časti sa nachádza legenda (Layers), ktorá umožňuje filtrovanie podľa zdroja dát a

závažnosti. Z pohľadu SOC architektúry tento dashboard predstavuje vizualizačnú vrstvu, ktorá dopĺňa analytické údaje o geografický kontext a podporuje identifikáciu pôvodu bezpečnostných incidentov.

2.9.2 Technické parametre a overenie funkčnosti

Pre nástroj Kibana sme pri nasadení definovali maximálnu veľkosť operačnej pamäte, s ktorou môže pracovať. Definovali sme 8 GB. Kibana nemá predvolene definované obmedzenie operačnej pamäte, pridelením obmedzenia sme zabezpečili, že Kibana nealokuje všetku pamäť pre svoje využitie. 8 GB sme zvolili pre poskytnutie dostatočných pamäťových zrojov pri zobrazovaní zložitých vizualizácií.

Kibana je schopná overiť TLS certifikát nástroja Elasticsearch, čo zabezpečuje šifrovanú komunikáciu. Elasticsearch si nástroj Kibana pri pripojení overuje pomocou mena a hesla, sú využité predvolené používateľské účty. Autentifikácia sa konfiguruje v hlavnom konfiguračnom súbore nástroja Kibana. Pre zvýšenie bezpečnosti nie je heslo predvoleného používateľa uložené v tzv. „plaintaxt“ podobe, ale pre uloženia hesla využívame keystore, obdobne ako aj pri ostatných nástrojoch. Sieťová komunikácie je zabezpečená pomocou nástroja iptables, ktorým sme definovali firewall pravidlá. Pre prihlásenie do UI nástroja kibana sa používajú používateľské účty s rolami na to určenými, tzv. riadenie prístupu na základe rolí (RBAC).

Kibana je monitorovaná nástrojmi Zabbix, LibreNMS a taktiež pomocou metricbeat. Takýmto monitorovaním sledujeme dostupnosť a stabilitu danej služby.

Služba Kibana taktiež beží vo svojom LXC kontajnery na rovnakom fyzickom servery ako zvyšok ELK staku. Kontajner má IP adresu 192.168.2.188, ktorá je dostupná iba zo špecifického IP rozsahu. V adresáry Kibana sa nachádza konfiguračný súbor, rovnako ako aj certifikáty s certifikačnou autoritou elasticsearch na zabezpečenie šifrovanej komunikácie. Ukážka konfiguračného súboru kibana.yml:

```
server.port: 5601
server.host: "0.0.0.0"
server.publicBaseUrl: "https://[interná doména]"
server.ssl.enabled: false
elasticsearch.hosts: ["https://192.168.2.178:9200"]
elasticsearch.username: "kibana_system"
elasticsearch.requestTimeout: 180000
elasticsearch.ssl.certificateAuthorities:                                [
"/etc/kibana/certs/kibana/elasticsearch-ca.pem" ]
logging:
  appenders:
    file:
      type: file
      fileName: /var/log/kibana/kibana.log
      layout:
        type: json
  root:
    appenders:
      - default
      - file
pid.file: /run/kibana/kibana.pid
xpack.encryptedSavedObjects.encryptionKey: [key1]
xpack.reporting.encryptionKey: [key2]
```

```
xpack.security.encryptionKey: [key3]
monitoring.kibana.collection.enabled: false
monitoring.ui.enabled: true
monitoring.ui.elasticsearch.hosts: ["https://192.168.2.178:9200"]
     monitoring.ui.ccs.enabled: false
```

2.9.3 Využitie Kibany v rámci PoC

V rámci PoC bola Kibana využívaná ako hlavný nástroj na vizualizáciu a analýzu bezpečnostných dát. Jej hlavný prínos spočíval v tom, že umožňovala prehľadne zobrazovať udalosti z rôznych zdrojov a transformovať ich do podoby použiteľnej pre analytikov SOC.

Praktické využitie bolo orientované najmä na:

- prezeranie logov pomocou modulu Discover,
- tvorbu analytických dashboardov,
- správu detekčných pravidiel a alertov,
- monitorovanie stavu ELK stacku.

V rámci nasadenia boli vytvorené viaceré dashboardy, napríklad:

- IDS DMZ – korelácia detekcií zo Suricata a Fortigate,
- Login Activity Monitoring – sledovanie prihlasovacích udalostí,
- Zaťaženie Stacku – monitoring výkonu ELK komponentov.

Tieto dashboardy umožňujú identifikovať anomálie, sledovať správanie používateľov a vyhodnocovať bezpečnostný stav infraštruktúry.

Kibana tak v tejto etape neplnila úlohu detekcie, ale poskytovala analytické a vizualizačné nástroje potrebné pre efektívnu interpretáciu dát.

2.10 Wiki.js

2.10.1 Úloha Wiki.js v PoC

Wiki.js je v rámci PoC riešenia využitý ako centrálny nástroj pre správu znalostnej bázy a internej dokumentácie. V navrhovanom SOC riešení neplní úlohu detekčného ani reakčného komponentu, ale predstavuje znalostnú a metodickú vrstvu, ktorá podporuje jednotný prístup k dokumentácii, postupom a interným smerniciam. Jeho význam spočíva najmä v tom, že umožňuje systematicky zhromažďovať a spravovať runbooky, playbooky a prevádzkové postupy SOC tímu.

V rámci PoC je Wiki.js chápaný ako nástroj, ktorý dopĺňa technické komponenty riešenia o dokumentačný kontext. Pri riešení bezpečnostných udalostí tak umožňuje analytikom rýchly prístup k definovaným postupom, odporúčaniam a interným pravidlám. Zároveň podporuje onboarding nových členov tímu a znižuje závislosť na individuálnych znalostiach.

Wiki.js v SOC poskytuje jednotné miesto pre evidenciu a zdieľanie znalostí. V tomto zmysle predstavuje doplnkový, no dôležitý komponent riešenia, ktorý zvyšuje konzistentnosť postupov a vytvára základ pre ďalšie rozšírenie dokumentačných a integračných scenárov.

2.10.2 Nasadenie a základná konfigurácia Wiki.js

V rámci PoC riešenia bol Wiki.js nasadený ako samostatná webová aplikácia určená pre správu dokumentácie a znalostí. Z pohľadu implementácie bol systém prevádzkovaný kontajnerizačne s využitím Docker prostredia a relačnej databázy (PostgreSQL), čo umožňuje jednoduché nasadenie, správu a opakovateľnosť inštalácie.

Základná konfigurácia bola zameraná na sprístupnenie webového rozhrania a vytvorenie základnej štruktúry znalostnej bázy. Systém bol pripravený na správu používateľov, rolí a prístupových práv, pričom databázová vrstva zabezpečuje ukladanie obsahu a jeho verzií.

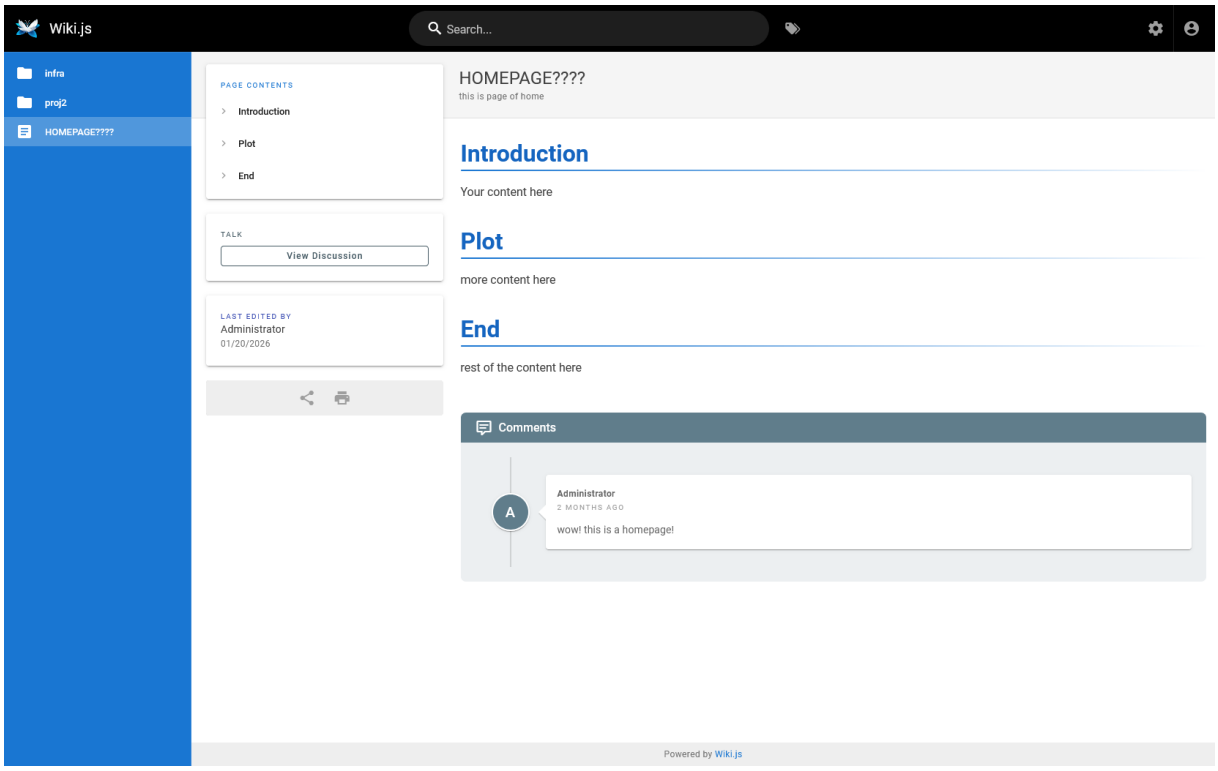
V rámci PoC boli do systému zavedené základné dokumentačné objekty, najmä runbooky, playbooky a interné postupy, ktoré slúžia ako referenčný materiál pri prevádzke SOC.

2.10.3 Využitie Wiki.js v rámci PoC

V rámci PoC bol Wiki.js využívaný ako znalostná a dokumentačná vrstva pre podporu prevádzky SOC. Jeho hlavný prínos spočíval v tom, že umožňoval centralizované uchovávanie a sprístupnenie dokumentácie potrebnej pri riešení incidentov a prevádzkových úloh.

Praktické využitie v tejto etape bolo orientované najmä na tvorbu a správu základných dokumentov, ako sú operačné postupy a metodiky. Wiki.js tak neplnil úlohu technického bezpečnostného nástroja, ale dopĺňal riešenie o znalostný kontext potrebný pre efektívnu prácu analytikov.

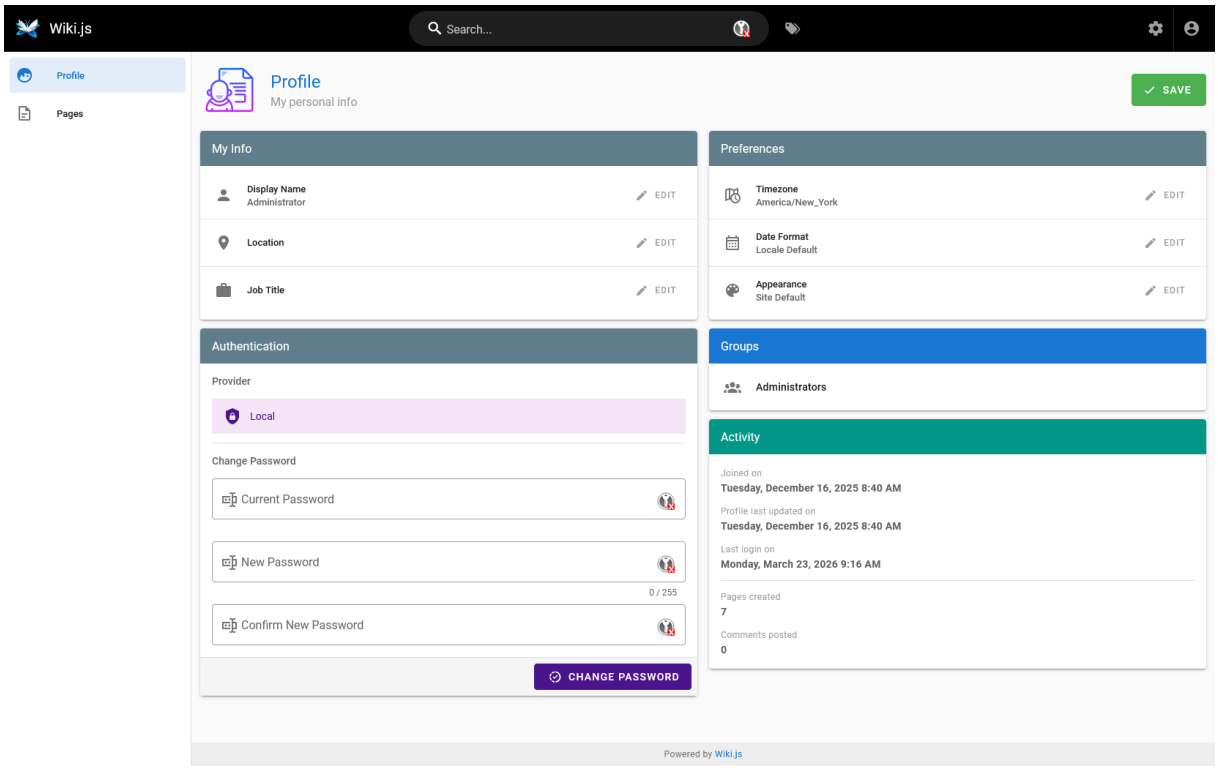
V tomto rozsahu jeho nasadenie potvrdilo, že ide o vhodný doplnok PoC riešenia a o použiteľný základ pre ďalšie rozšírenie, najmä v oblasti prepojenia na procesy incident response a automatizácie.



Obrázok zobrazuje webové rozhranie nástroja Wiki.js, ktorý je v rámci SOC využívaný ako centrálna platforma pre správu dokumentácie a znalostí.

V ľavej časti sa nachádza navigačný panel so štruktúrou stránok a obsahom dokumentu, ktorý umožňuje rýchly pohyb medzi jednotlivými sekciami. Centrálna časť zobrazuje konkrétnu stránku rozdelenú do logických častí (napr. Introduction, Plot, End), čo ilustruje spôsob tvorby runbookov a prevádzkových postupov.

Spodná časť obsahuje sekciu komentárov, ktorá podporuje spoluprácu a priebežné dopĺňanie informácií. Z pohľadu SOC architektúry Wiki.js predstavuje znalostnú vrstvu, ktorá zabezpečuje centralizované uchovávanie postupov a podporuje konzistentné riešenie incidentov.

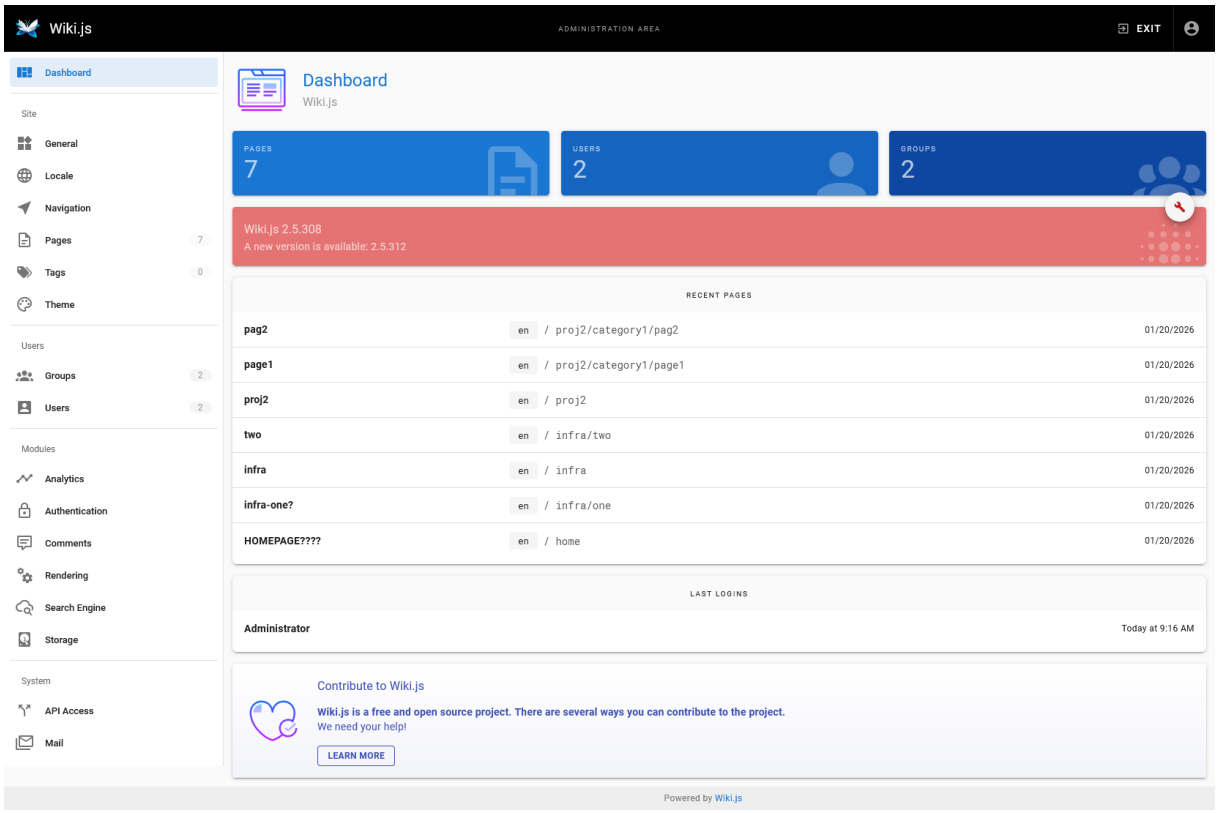


Obrázok zobrazuje používateľské rozhranie nástroja Wiki.js v sekcii správy používateľského profilu.

V centrálnej časti sa nachádzajú informácie o používateľovi (napr. zobrazované meno, lokalita, pracovná pozícia) a konfiguračné nastavenia, ako časové pásmo, formát dátumu alebo vzhľad rozhrania. Súčasťou je aj sekcia autentifikácie, ktorá umožňuje správu prihlasovacích údajov vrátane zmeny hesla.

Na pravej strane sú zobrazené informácie o používateľských skupinách (napr. Administrators) a aktivite, vrátane údajov o prihláseniach a počte vytvorených stránok.

Z pohľadu SOC architektúry obrázok ilustruje správu používateľov a prístupových práv v rámci Wiki.js, čo je dôležité pre riadenie prístupu k internej dokumentácii a zabezpečenie kontrolovaného zdieľania znalostí v rámci tímu.

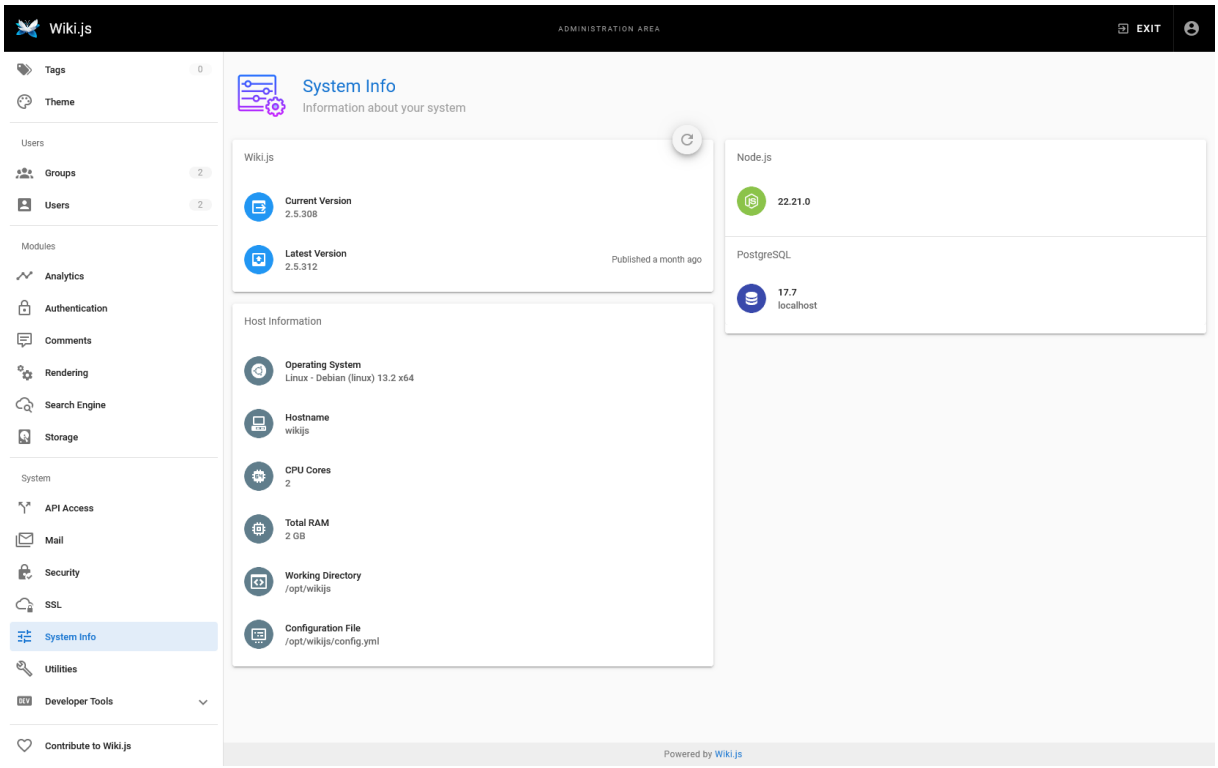


Obrázok zobrazuje administračný dashboard nástroja Wiki.js, ktorý poskytuje prehľad o stave znalostnej bázy a základných systémových metrikách.

V hornej časti sa nachádzajú sumarizačné panely s informáciami o počte stránok, používateľov a skupín, čo ilustruje rozsah evidovanej dokumentácie a používateľskej základne. Centrálna časť obsahuje zoznam naposledy upravených alebo vytvorených stránok, ktoré predstavujú jednotlivé dokumentačné záznamy (napr. runbooky alebo interné postupy).

V ľavej časti je dostupné administračné menu umožňujúce správu obsahu, používateľov, autentifikácie a ďalších systémových nastavení.

Z pohľadu SOC architektúry obrázok ilustruje Wiki.js ako znalostnú vrstvu, ktorá centralizuje dokumentáciu a poskytuje prehľad o jej stave a využívaní, čím podporuje efektívnu správu a dostupnosť znalostí pre SOC tím.



Obrázok zobrazuje administračné rozhranie nástroja Wiki.js v sekcii „System Info“, ktorá slúži na prehľad systémových a prevádzkových informácií o nasadení.

V centrálnej časti sú zobrazené informácie o verzii aplikácie Wiki.js (aktuálna a dostupná verzia), ako aj údaje o hostiteľskom prostredí, vrátane operačného systému, hostname, počtu CPU jadier, veľkosti pamäte a umiestnenia konfiguračných súborov. Na pravej strane sú uvedené technológie, na ktorých je systém postavený, konkrétne verzia Node.js a databázy PostgreSQL.

V ľavej časti sa nachádza administračné menu, ktoré umožňuje správu používateľov, autentifikácie, vyhľadávania, úložiska a ďalších systémových nastavení.

Z pohľadu SOC architektúry obrázok ilustruje technické zázemie a správu platformy Wiki.js, ktorá zabezpečuje spoľahlivú prevádzku znalostnej bázy a dostupnosť dokumentácie pre členov SOC tímu.

2.10.4 Aktuálny stav a obmedzenia Wiki.js v SOC

Wiki.js bol v rámci PoC úspešne nasadený a základne nakonfigurovaný ako funkčný nástroj pre správu internej dokumentácie. V tomto stave poskytuje použiteľný základ pre evidenciu znalostí, správu runbookov a podporu prevádzkových procesov SOC. Z pohľadu PoC sa potvrdilo, že ide o vhodný komponent na doplnenie riešenia o znalostnú a metodickú vrstvu.

Na druhej strane je potrebné uviesť, že jeho využitie v PoC ostalo na úrovni základnej dokumentácie bez širších integračných väzieb. Neboli implementované pokročilé scenáre prepojenia na ostatné komponenty SOC (napr. automatické referencovanie incidentov alebo integrácia so SIEM/SOAR), a nástroj tak v tejto etape funguje primárne ako samostatný dokumentačný systém.

Z tohto dôvodu je Wiki.js v PoC potrebné chápať ako funkčný, ale zatiaľ nie plne rozvinutý komponent. Napriek tomu jeho nasadenie preukázalo praktickú využiteľnosť a potvrdilo vhodnosť jeho zaradenia do širšej architektúry SOC služby.

Plus v zvyšnej dokumentácii je aj nejaký screen z nastavenia emailov alebo čosi také myslím...

3 ZÁVER

Predmetom tohto dokumentu bolo zdokumentovanie funkčného PoC (Proof of Concept) riešenia SOCaaS platformy INNOSOC, vytvoreného v rámci pracovného balíka KPB5 – Realizácia riešenia. Dokument zachytáva reálny stav implementácie navrhovaných komponentov, ich nasadenie, konfiguráciu a základné overenie funkčnosti v experimentálnom prostredí.

3.1 Súhrn realizovaných aktivít

V rámci PoC bola realizovaná implementácia nasledujúcich komponentov navrhovanej architektúry SOCaaS:

ELK Stack (SIEM) – nasadený ako centrálny systém pre zber, ukladanie, analýzu a vizualizáciu bezpečnostných udalostí. Elasticsearch cluster pozostáva zo siedmich uzlov s rozdelením podľa dátových úrovní (hot, warm, cold), pričom Logstash zabezpečuje príjem a normalizáciu logov a Kibana poskytuje analytické a vizualizačné rozhranie vrátane vlastných dashboardov.

Wazuh – nasadený ako EDR/SIEM agent pre monitorovanie koncových zariadení, detekciu bezpečnostných udalostí, hodnotenie konfigurácie (SCA), detekciu zraniteľností a monitorovanie integrity súborov (FIM). Integrácia do prostredia bola doplnená o notifikačné väzby na MS Teams.

Shuffle (SOAR) – nasadený ako nástroj pre orchestráciu a automatizáciu bezpečnostných reakcií. V rámci PoC boli overené základné integračné väzby s komponentmi ELK, DFIR-IRIS a OpenCTI.

OpenCTI – nasadený ako centrálna platforma pre správu informácií o kybernetických hrozbách (CTI). Platforma bola prepojená s externými zdrojmi dát (Abuse.ch, MITRE ATT&CK, Ransomwarelive) a s komponentom Shuffle pre automatizáciu toku CTI dát.

DFIR-IRIS – nasadený ako nástroj pre evidenciu a správu bezpečnostných incidentov. Overená bola centralizácia alertov, incident management, notifikačný systém a prepojenie so Shuffle.

Greenbone Community Edition – nasadená ako nástroj pre skenovanie a správu zraniteľností. V rámci PoC bol overený základný sken a odosielanie logov do ELK Stacku cez Filebeat.

GLPI – nasadený ako CMDB a ITSM nástroj pre evidenciu IT aktív, správu incidentov a mapovanie organizačného kontextu na technické bezpečnostné udalosti.

NetBox – nasadený ako nástroj pre evidenciu sieťovej infraštruktúry, správu IP priestoru a topologické zobrazenie väzieb medzi zariadeniami.

Wiki.js – nasadený ako znalostná báza pre správu dokumentácie, runbookov a playbookov potrebných pri prevádzke SOC.

Kibana – v rámci ELK Stacku boli vytvorené vlastné dashboardy pre vizualizáciu bezpečnostných dát (IDS DMZ, Login Activity, Zaťaženie Stacku) a overená bezpečnostná konfigurácia vrátane TLS, RBAC a retenčnej politiky (ILM).

3.2 Charakter a rozsah PoC

Funkčný PoC bol vytvorený ako súbor samostatne nainštalovaných a nakonfigurovaných inštancií jednotlivých komponentov v prostredí OpenStack. Každý komponent bol nasadený, základne nakonfigurovaný a funkčne overený ako samostatná služba. Tento prístup umožnil overiť technickú realizovateľnosť navrhovanej architektúry a základnú funkčnosť každého komponentu v izolovanom prostredí.

V rámci PoC boli zároveň načrtnuté a čiastočne overené reálne možnosti integrácie medzi jednotlivými nástrojmi. Medzi overené integračné väzby patria najmä:

tok bezpečnostných udalostí z monitorovaných zariadení cez Wazuh a Filebeat do ELK Stacku, prepojenie ELK Stacku so Shuffle pre orchestráciu reakcií na detekované udalosti, import alertov z Wazuh do DFIR-IRIS pre centralizovanú evidenciu incidentov, distribúcia CTI dát z OpenCTI do ďalších komponentov prostredníctvom Shuffle, odosielanie notifikácií z Wazuh do MS Teams cez Incoming Webhook.

Nie všetky plánované integračné väzby boli v čase spracovania tohto výstupu dotiahnuté do finálnej úrovne funkčnosti. V niektorých prípadoch boli komponenty úspešne nasadené samostatne, avšak ich hlbšia integrácia zostala len čiastočne realizovaná alebo overená na konceptuálnej úrovni. PoC tak zachytáva reálny stav implementácie, nie finálny stav produkčnej architektúry.

3.3 Overenie a testovanie

Funkčné overenie realizovaného PoC je predmetom samostatného výstupu V17 – Výsledky a vyhodnotenie testov (KPB5), ktorý obsahuje:

experimentálne testy metód strojového učenia a umelej inteligencie pre detekciu bezpečnostných hrozieb (spam, DoS/DDoS, malware, phishing, APT), overenie detekčných schopností nasadeného SIEM prostredníctvom simulovaného útoku podľa modelu Cyber Kill Chain, automatické testovanie detekčných pravidiel pomocou Atomic Red Team, manuálne testovanie korelačných pravidiel v Kibane, záťažové testovanie nasadeného ELK Stacku, integračné testy medzi komponentmi vrátane DFIR-IRIS, Shuffle a OpenCTI, funkčné testy jednotlivých komponentov (Wazuh, Greenbone, GLPI, Wiki.js, ELK Stack, NetBox).

Výsledky testovania preukázali, že navrhované riešenie je vo svojom aktuálnom rozsahu funkčné a prevádzkovo spôsobilé. Podrobné výsledky, identifikátory testov a ich vyhodnotenie sú uvedené v dokumente V17.

3.4 Záverečné zhodnotenie

Realizovaný PoC preukázal technickú realizovateľnosť navrhovanej architektúry multitenantného operačného centra kybernetickej bezpečnosti riešeného ako otvorená cloudová služba. Všetky kľúčové komponenty boli úspešne nasadené, nakonfigurované a funkčne overené v experimentálnom prostredí.

Z hľadiska ďalšieho rozvoja riešenia PoC poskytuje dostatočný základ pre prechod do fázy rozšírenej integrácie a pilotnej prevádzky. Medzi hlavné oblasti ďalšieho rozvoja patria predovšetkým prehĺbenie integračných väzieb medzi komponentmi, rozšírenie automatizácie bezpečnostných reakcií prostredníctvom SOAR platformy, nasadenie ďalších zdrojov CTI dát a optimalizácia detekčných a korelačných pravidiel na základe poznatkov z testovania.

PoC ako celok potvrdzuje, že zvolený architektonický prístup založený na otvorených technológiách je životaschopný a poskytuje flexibilný základ pre budovanie komerčnej SOCaaS služby prispôsobiteľnej potrebám rôznych organizácií.