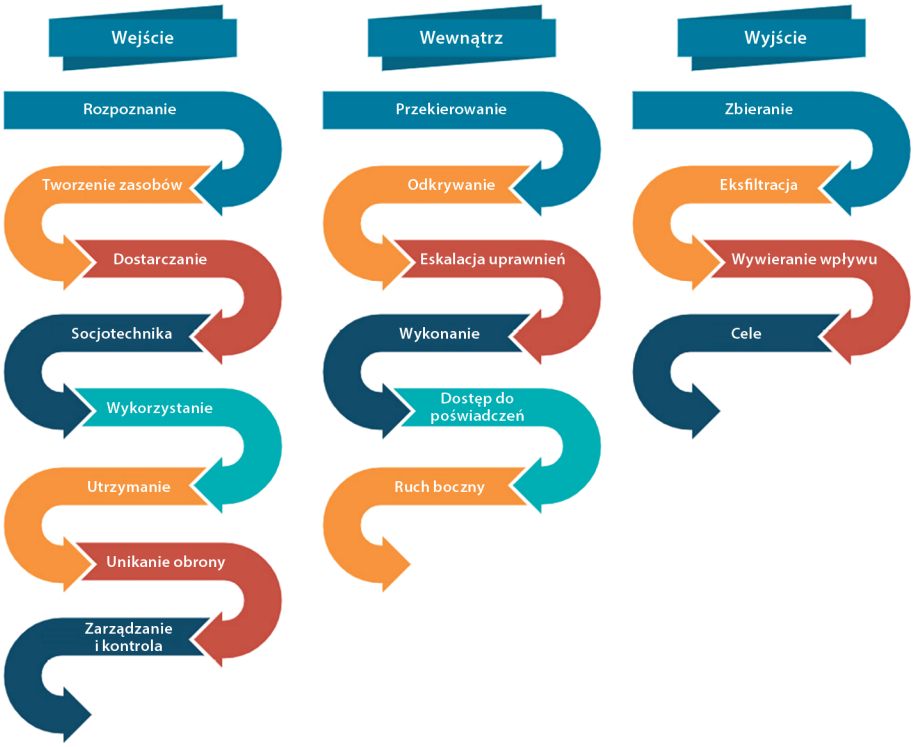
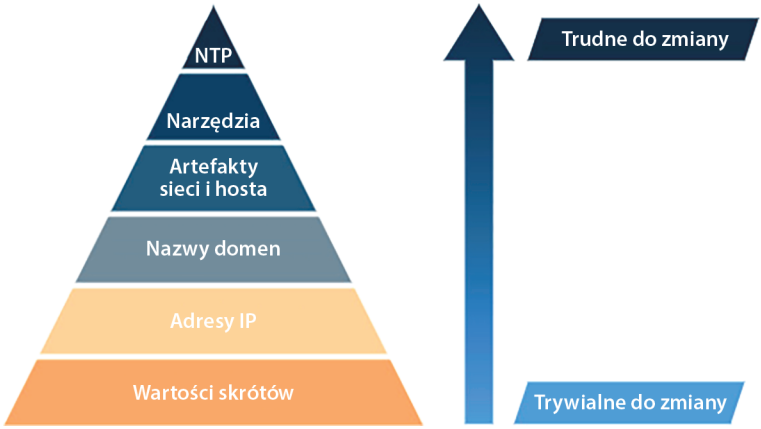
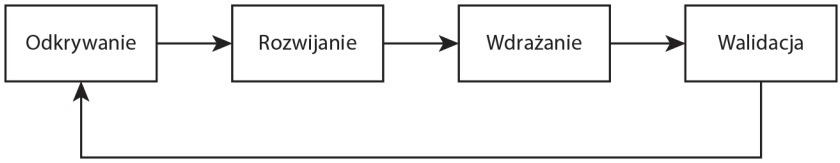
Rozdział 1. Podstawy inżynierii detekcji



Rysunek 1.1. Model Unified Kill Chain

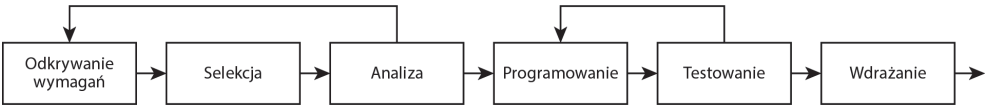


Rysunek 1.2. Model piramidy bólu Davida Bianco

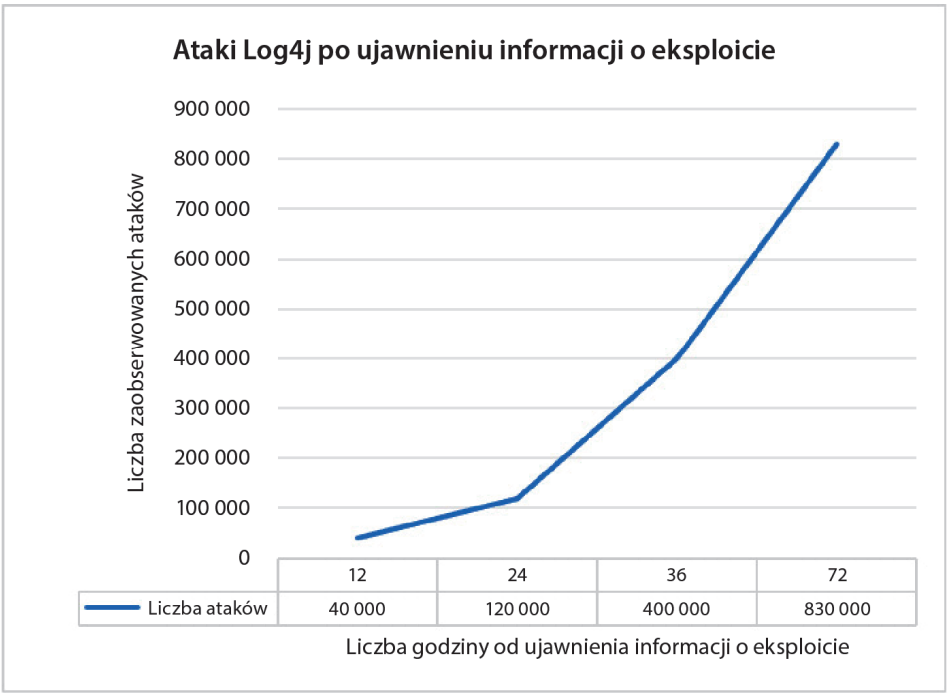


Rysunek 1.3. Procesy wchodzące w skład inżynierii detekcji

Rozdział 2. Cykl życia inżynierii detekcji

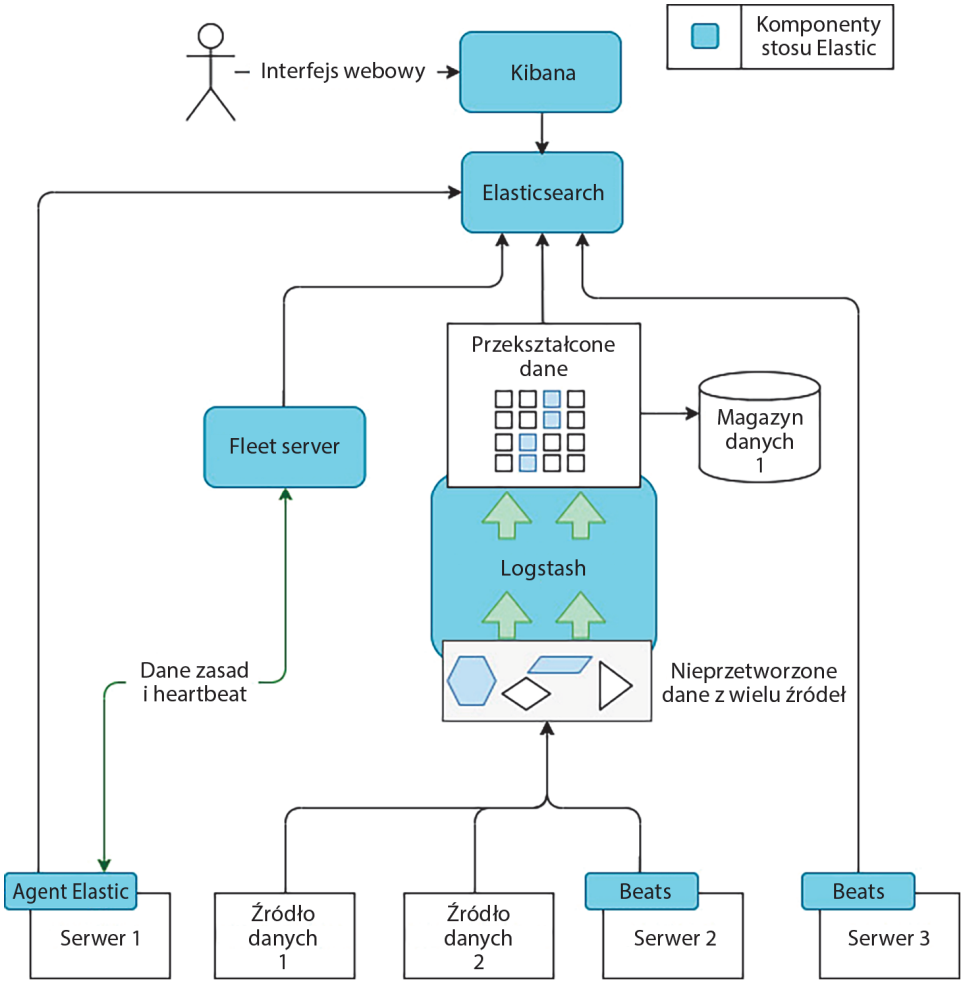


Rysunek 2.1. Cykl życia inżynierii detekcji

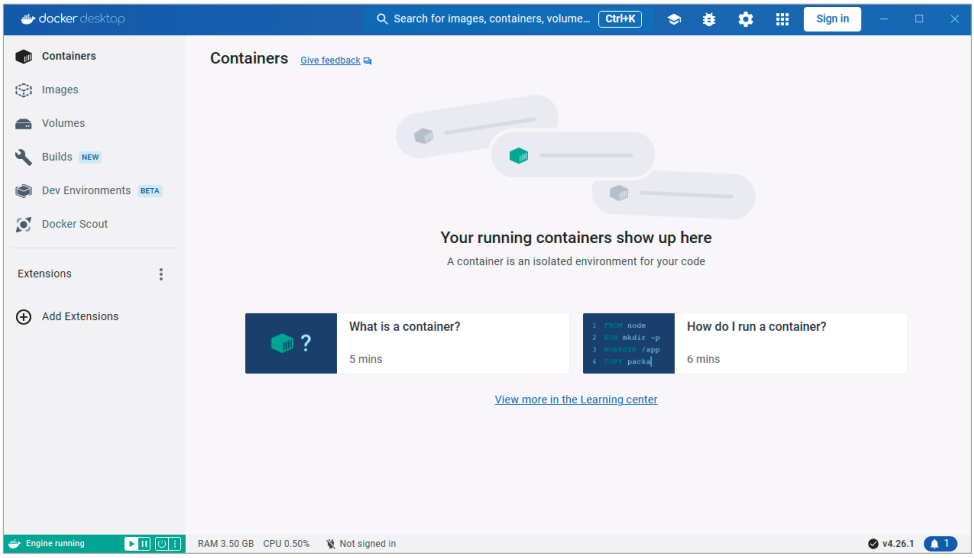


Rysunek 2.2. Statystyki eksploitów Log4j. Źródło danych: https://blog.checkpoint.com/security/the-numbers-behind-a-cyber-pandemic-detailed-dive/

Rozdział 3. Budowa laboratorium testowego inżynierii detekcji



Rysunek 3.1. Przykład wdrożenia Elasticsearch



Rysunek 3.2. Docker Desktop w systemie Windows

```
examiner@datasrv02:~$ docker --help

Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

Options:
      --config string      Location of client config files (default "/home/
  -c, --context string     Name of the context to use to connect to the dae
                           "docker context use")
  -D, --debug              Enable debug mode
  -H, --host list          Daemon socket(s) to connect to
  -l, --log-level string   Set the logging level ("debug"|"info"|"warn"|"er
      --tls                Use TLS; implied by --tlsverify
      --tlscacert string   Trust certs signed only by this CA (default "/ho
      --tlscert string     Path to TLS certificate file (default "/home/exa
      --tlskey string      Path to TLS key file (default "/home/examiner/.d
      --tlsverify          Use TLS and verify the remote
  -v, --version            Print version information and quit

Management Commands:
  builder     Manage builds
  compose*    Docker Compose (Docker Inc., v2.7.0)
  config      Manage Docker configs
  container   Manage containers
```

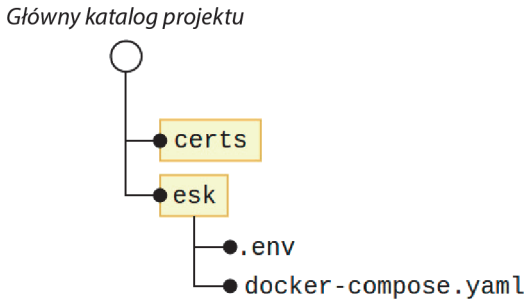
Rysunek 3.3. Pomoc systemu Docker

```
C:\Windows\system32>wsl -l
Podsystem Windows dla dystrybucji systemu Linux:
Ubuntu (domyślnie)
docker-desktop-data
docker-desktop
```

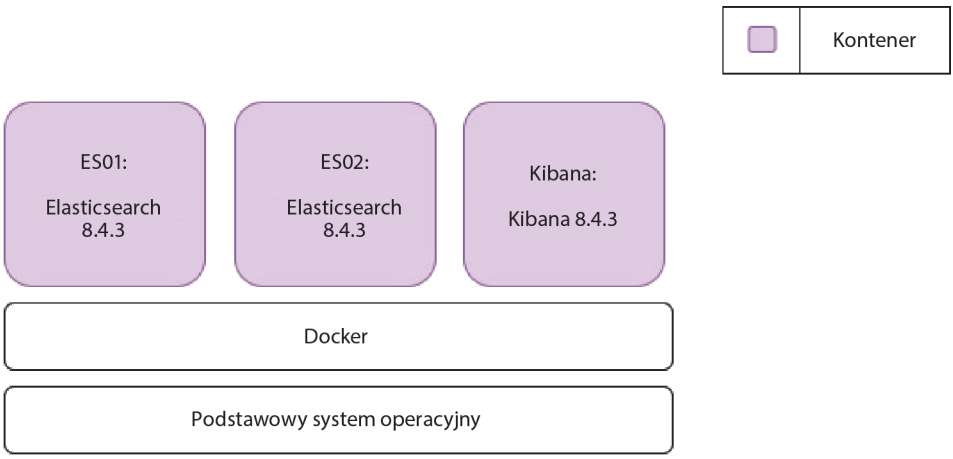
Rysunek 3.4. Lista dystrybucji wsl

```
C:\Windows\system32>wsl -d docker-desktop
EwelaRewela:/mnt/host/c/Windows/system32# sysctl -w vm.max_map_count=262144
vm.max_map_count = 262144
EwelaRewela:/mnt/host/c/Windows/system32#
```

Rysunek 3.5. Konfigurowanie pamięci wirtualnej Docker Desktop



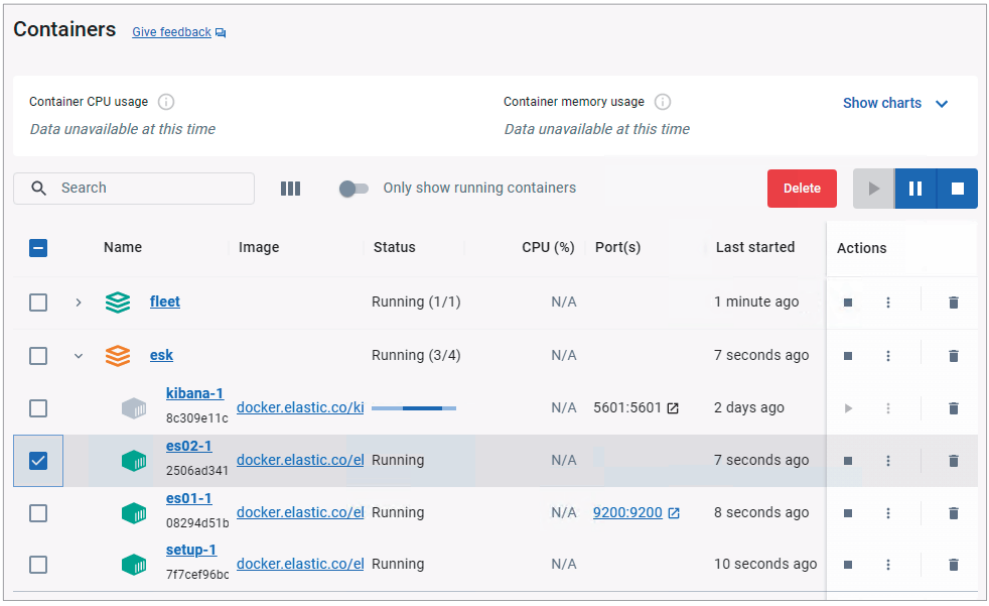
Rysunek 3.6. Struktura folderów projektu Elastic Stack



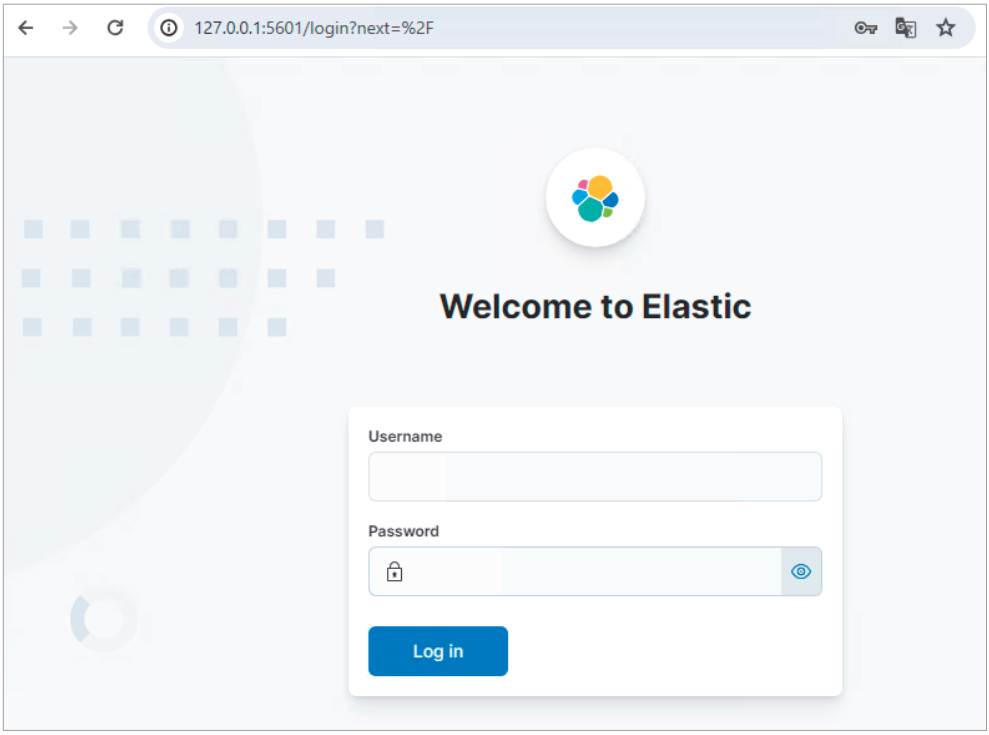
Rysunek 3.7. Wdrożenie platformy Docker

```
examiner@datasrv02:~/dev/delab/esk$ sudo docker compose ps
NAME            COMMAND                  SERVICE         STATUS              PORTS
esk-es01-1      "/bin/tini -- /usr/l…"   es01            running (healthy)   0.0.0.0:9200→9200/tcp, :::9200→9200/tcp, 9300/tcp
esk-es02-1      "/bin/tini -- /usr/l…"   es02            running (healthy)   9200/tcp, 9300/tcp
esk-kibana-1    "/bin/tini -- /usr/l…"   kibana          running (healthy)   0.0.0.0:5601→5601/tcp, :::5601→5601/tcp
esk-setup-1     "/bin/tini -- /usr/l…"   setup           exited (0)
```

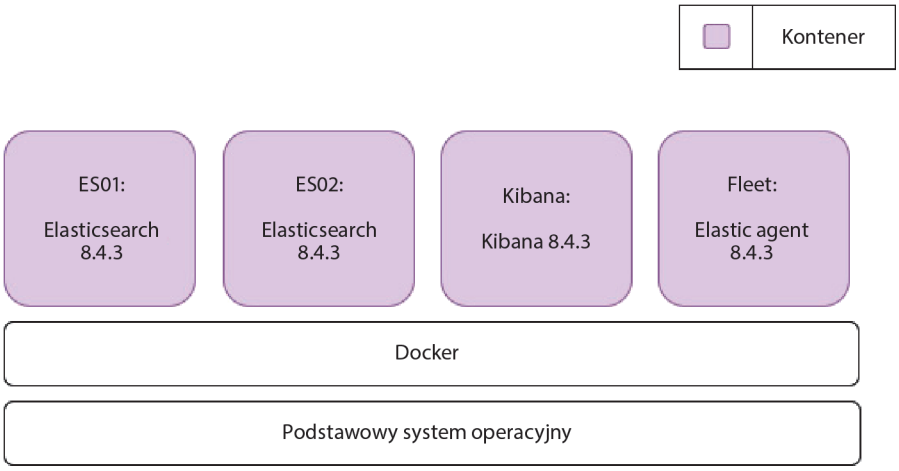
Rysunek 3.8. Status kontenera wyświetlany w terminalu



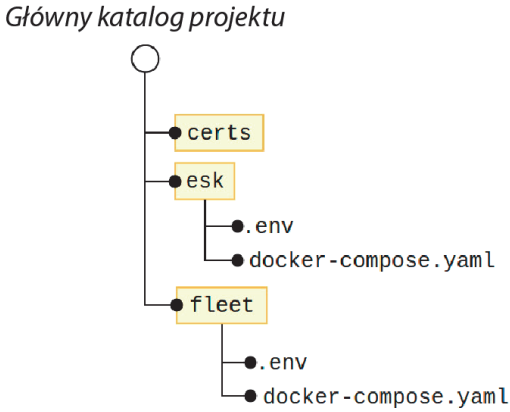
Rysunek 3.9. Status kontenerów w środowisku Docker Desktop



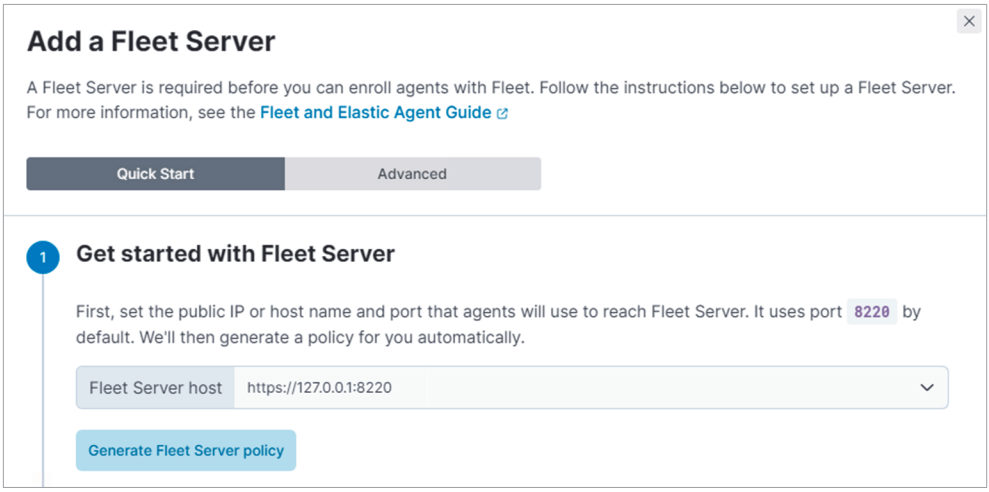
Rysunek 3.10. Strona logowania do systemu Elastic



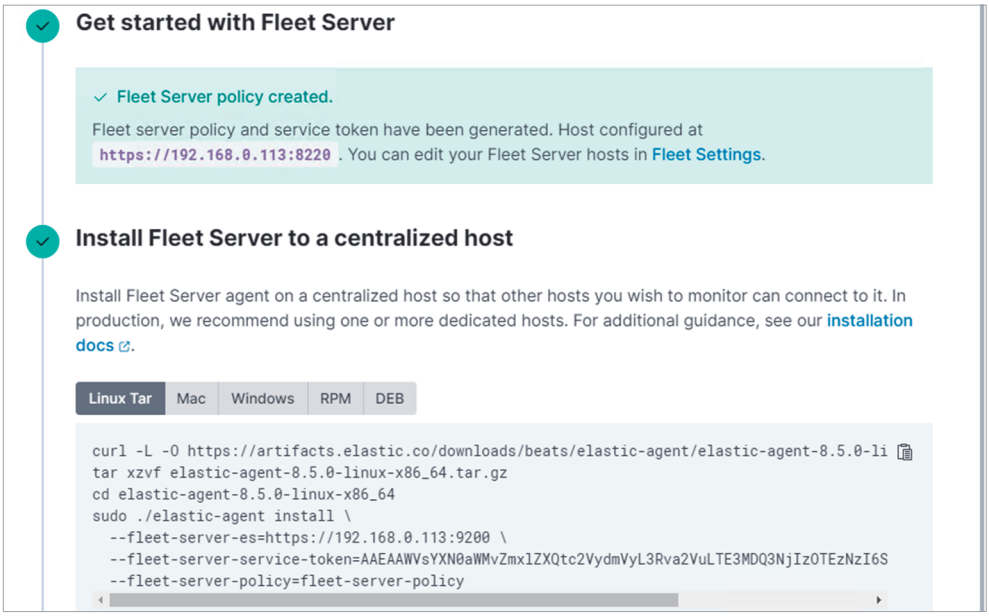
Rysunek 3.11. Wdrożenie platformy Docker z komponentem Fleet Server



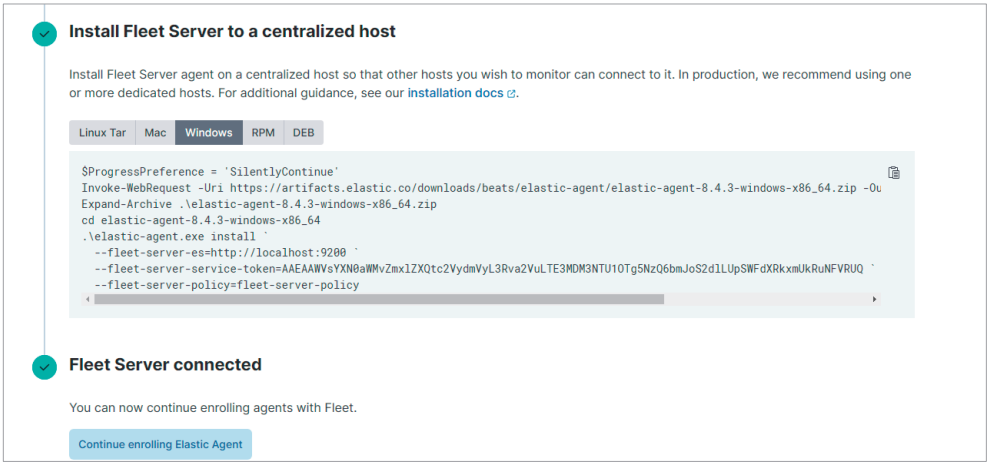
Rysunek 3.12. Struktura folderów projektu z komponentem Fleet



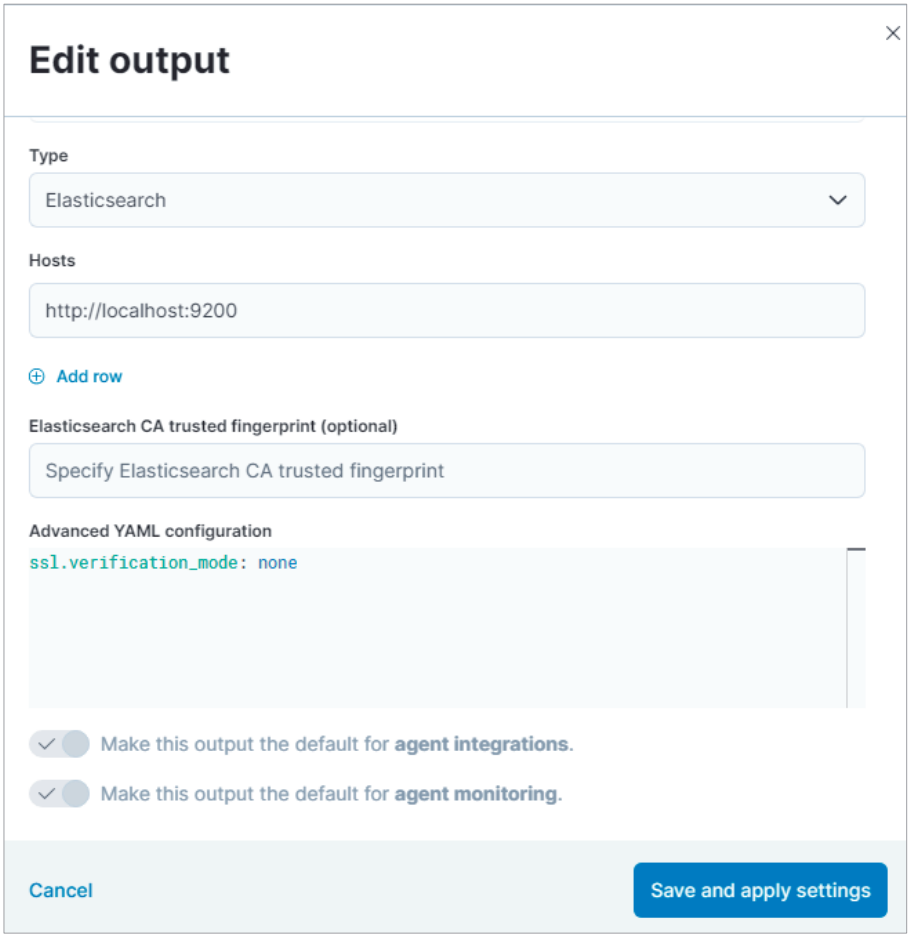
Rysunek 3.13. Ekran konfiguracji komponentu Fleet Server (krok 1.)



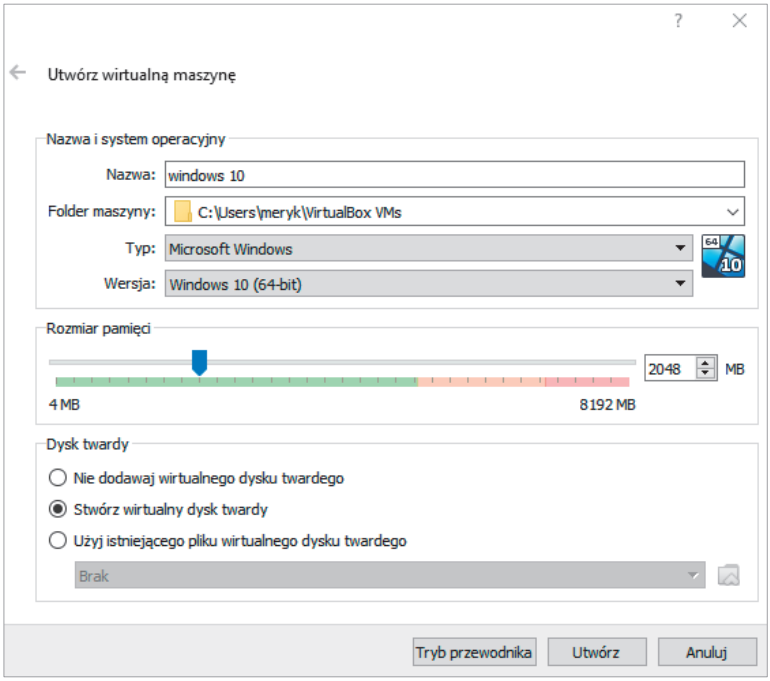
Rysunek 3.14. Ekran konfiguracji komponentu Fleet Server (krok 2.)



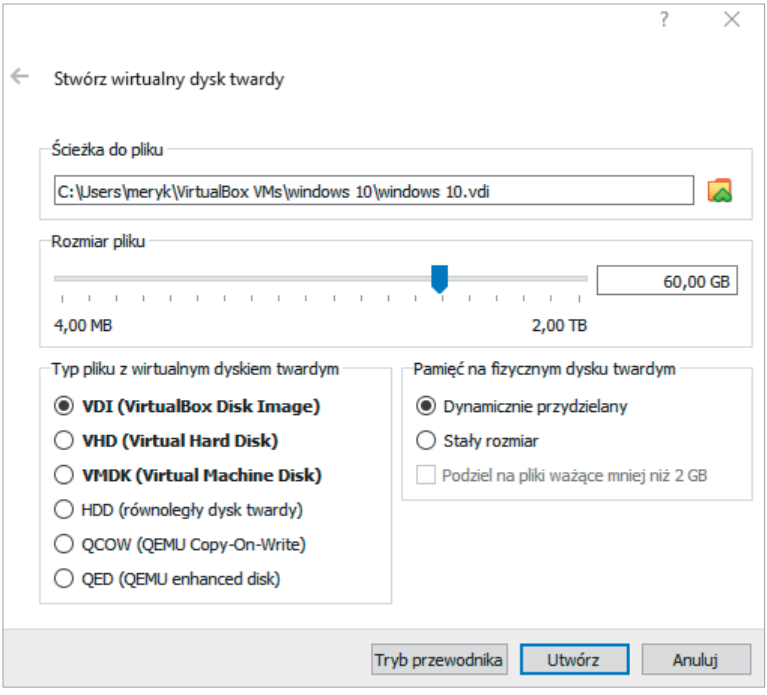
Rysunek 3.15. Ekran konfiguracji komponentu Fleet Server (krok 3.)



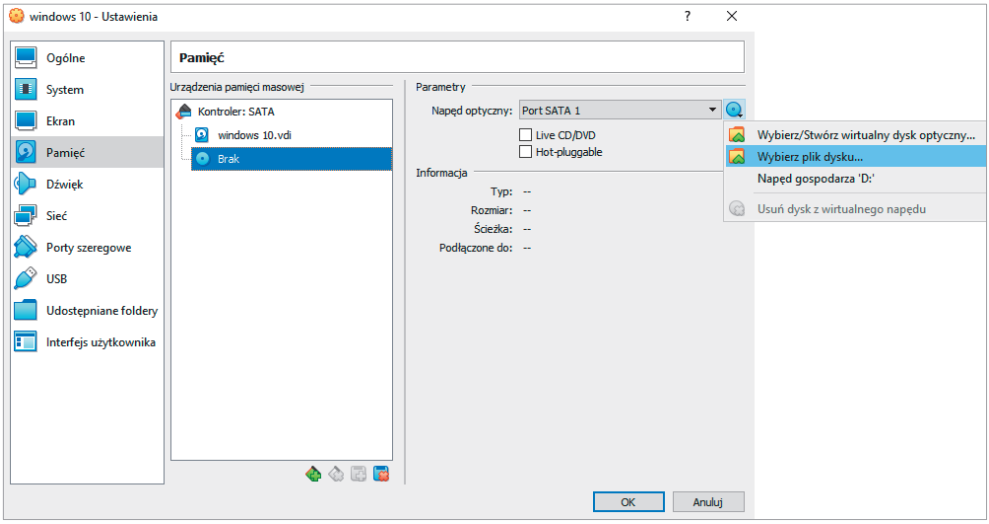
Rysunek 3.16. Konfiguracja wyjścia komponentu Fleet Server



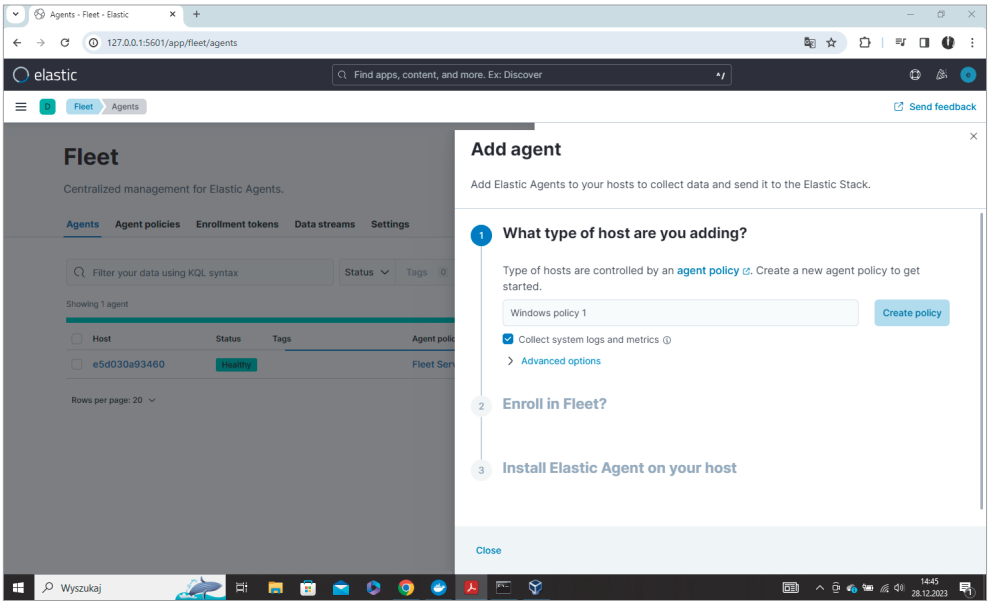
Rysunek 3.17. Obszar Create Virtual Machine w środowisku VirtualBox



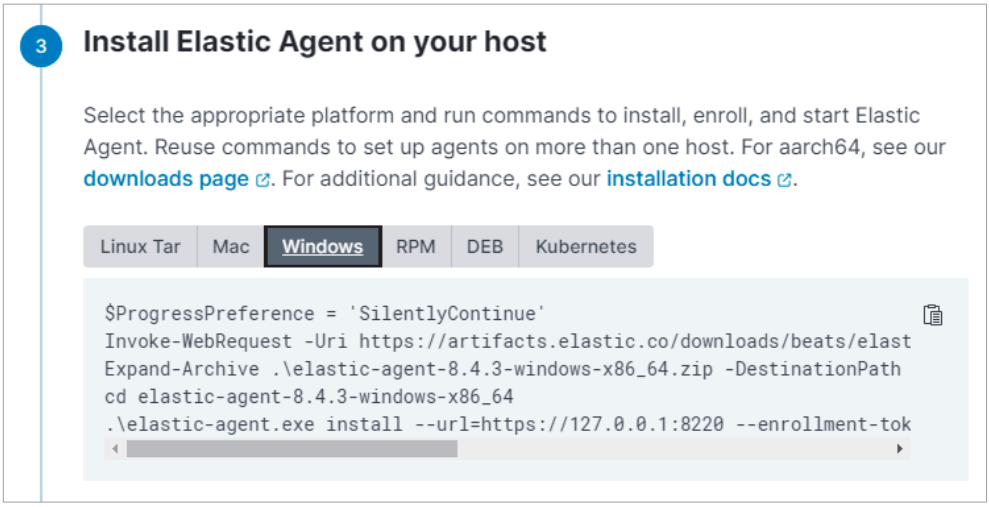
Rysunek 3.18. Obszar Create Virtual Hard Disk w środowisku VirtualBox



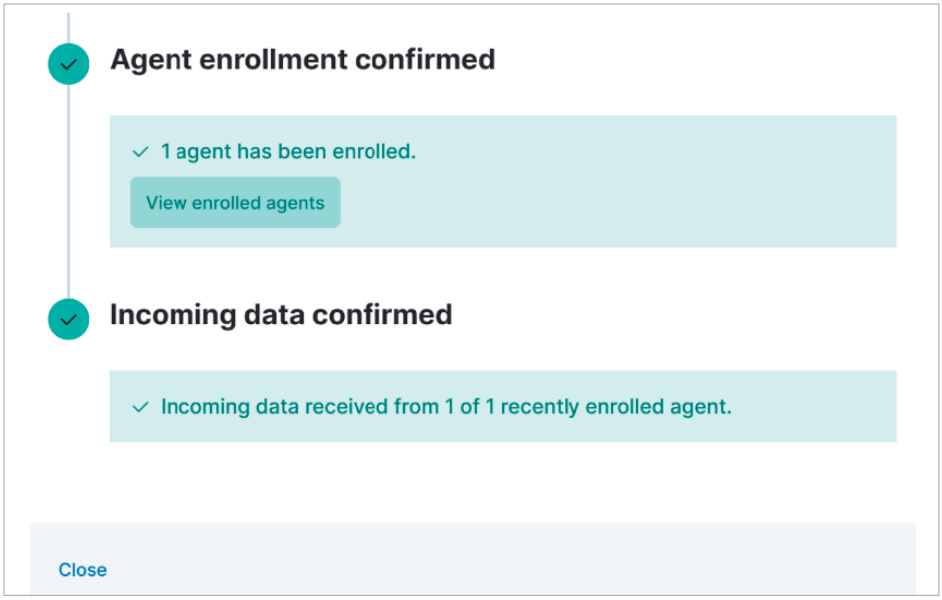
Rysunek 3.19. Ustawienia maszyny wirtualnej w środowisku VirtualBox



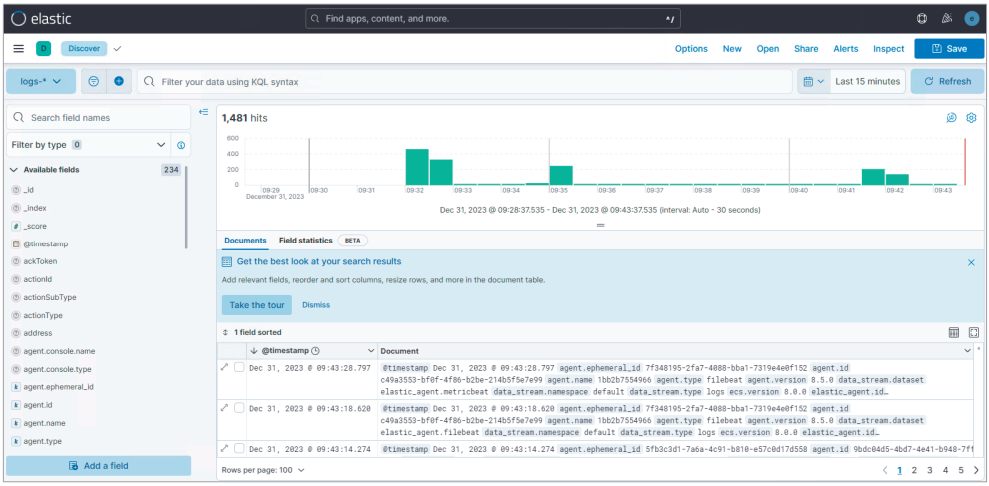
Rysunek 3.20. Dodawanie agenta



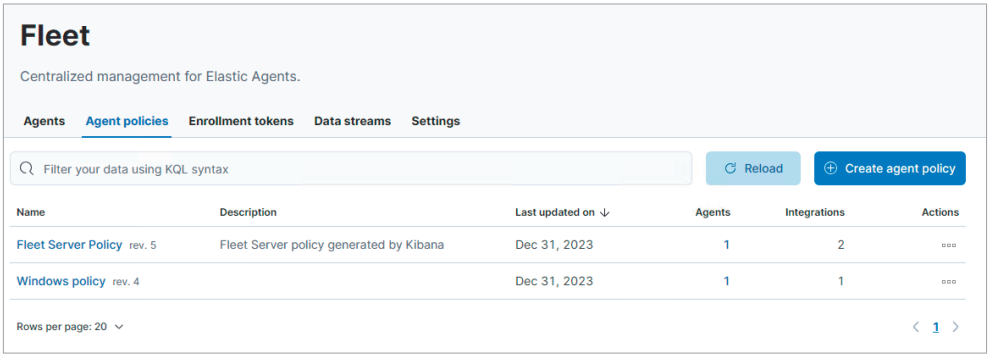
Rysunek 3.21. Skrypt instalacyjny Elastic Agent



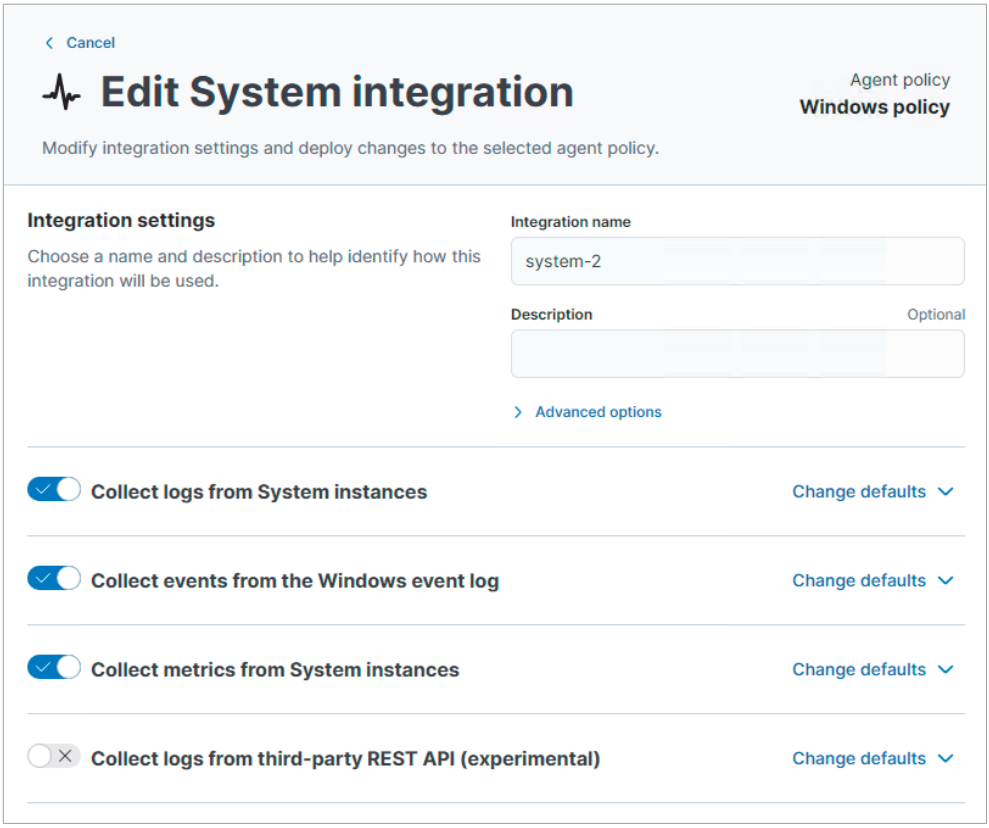
Rysunek 3.22. Potwierdzenie rejestracji agenta



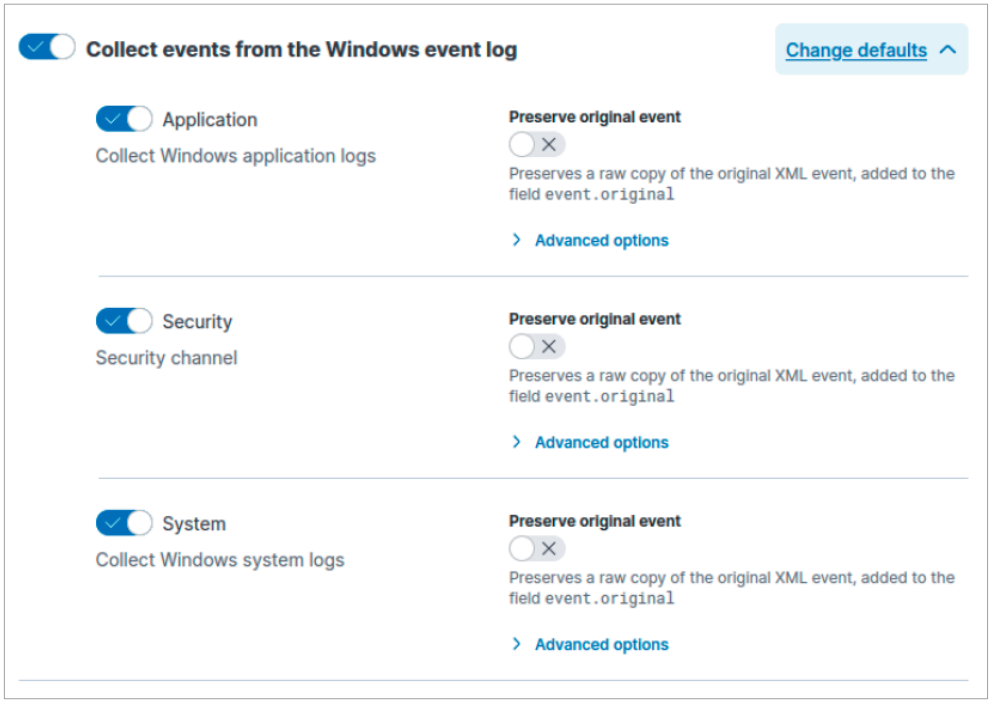
Rysunek 3.23. Widok Discover w systemie Elastic Stack



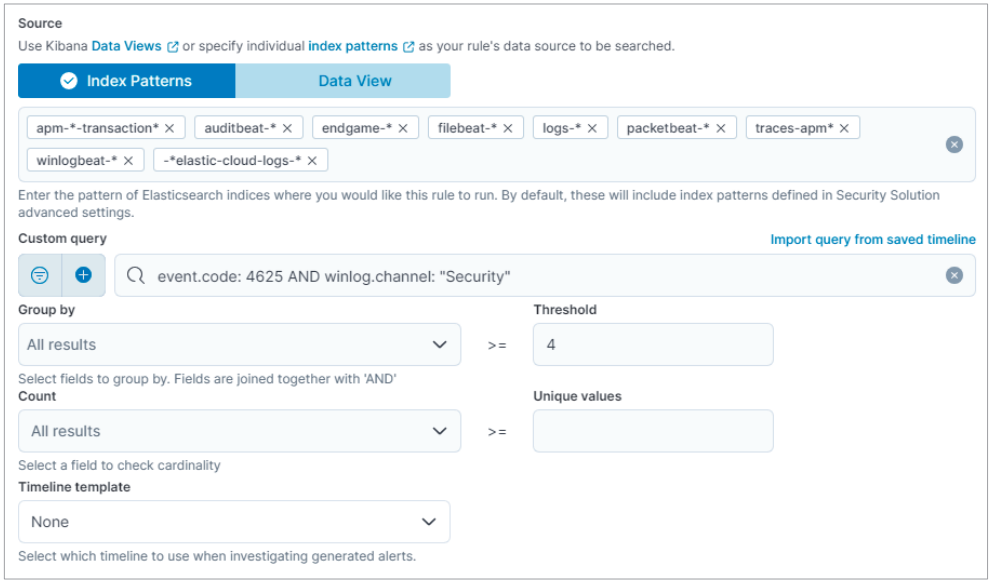
Rysunek 3.24. Zasady działania komponentu Elastic Agent



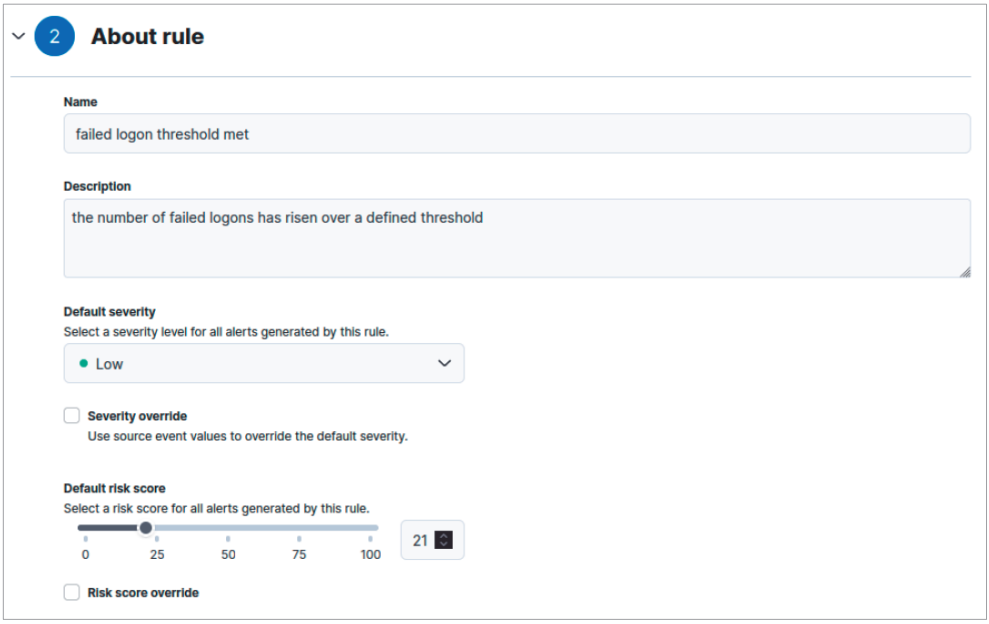
Rysunek 3.25. Integracja edycji systemu (Edit system)



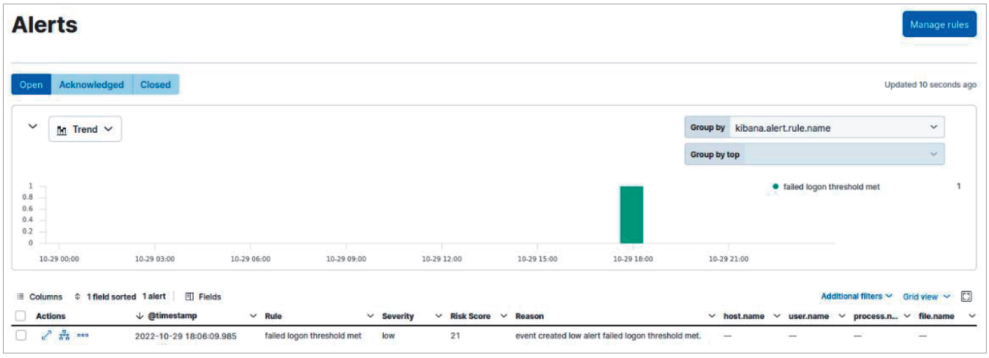
Rysunek 3.26. Ustawienia zbierania logów zdarzeń systemu Windows



Rysunek 3.27. Konfiguracja nowej reguły

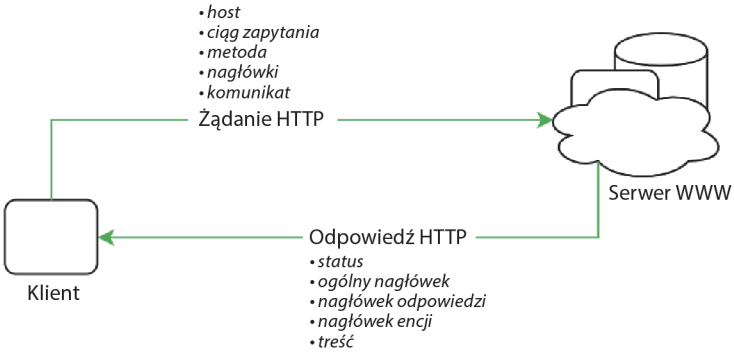


Rysunek 3.28. Szczegółowe dane nowej reguły

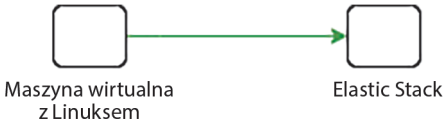


Rysunek 3.29. Pomyślne wykrycie zagrożenia przez mechanizm detekcji

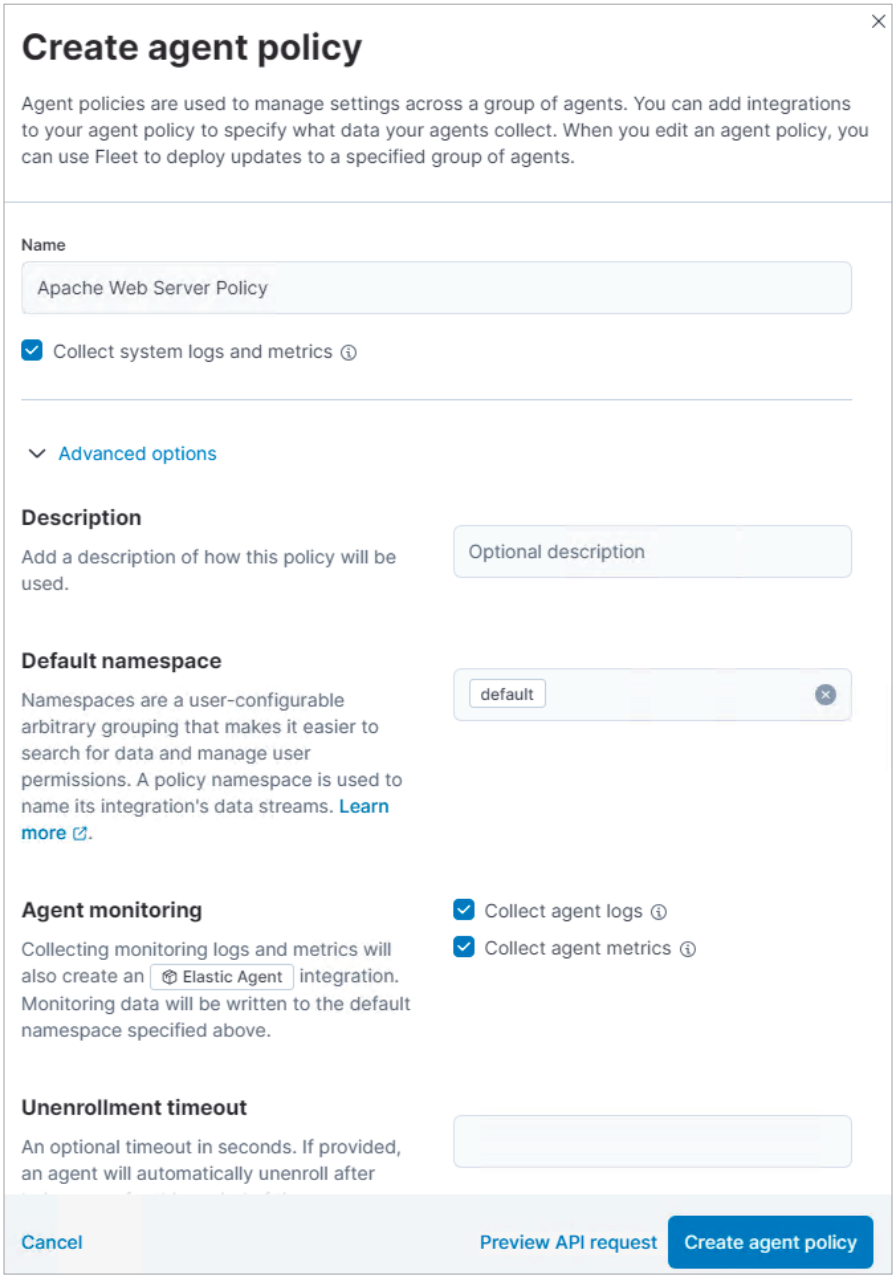
Rozdział 4. Źródła danych inżynierii detekcji



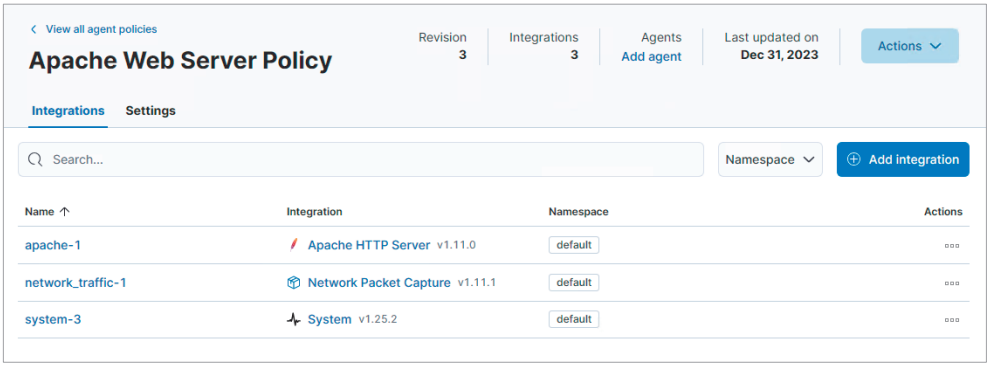
Rysunek 4.1. Przegląd ruchu na serwerze WWW



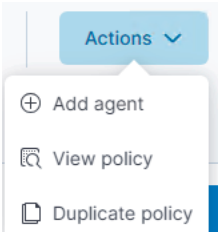
Rysunek 4.2. Instalacja z serwerem WWW



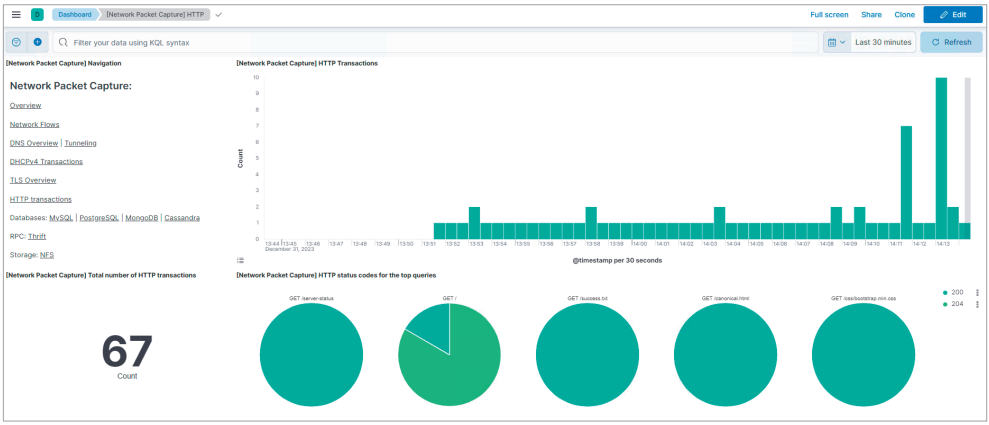
Rysunek 4.3. Tworzenie zasady agenta



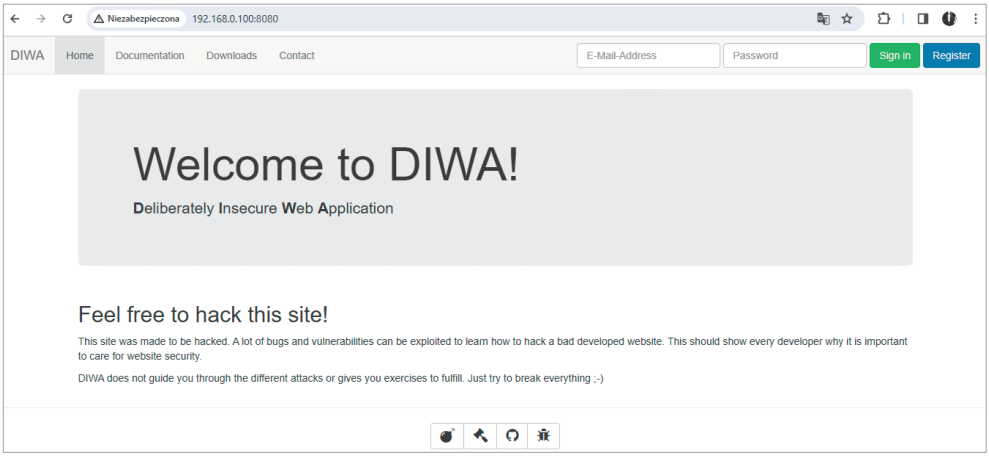
Rysunek 4.4. Zasady serwera WWW Apache



Rysunek 4.5. Dodawanie agenta Elastic

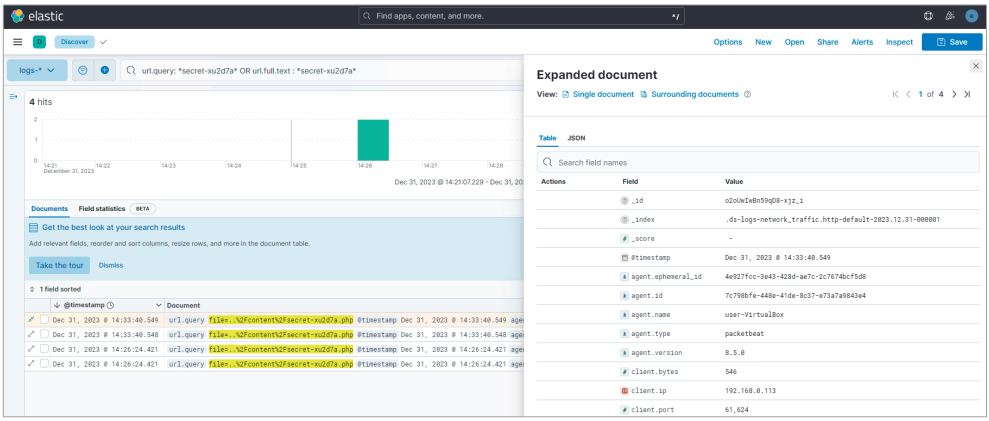


Rysunek 4.6. Pulpit nawigacyjny Network Packet Capture HTTP

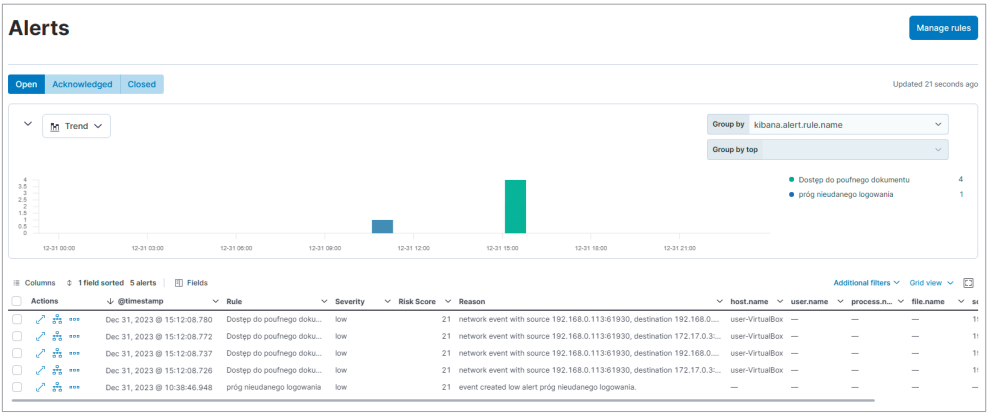


Rysunek 4.7. Główna strona aplikacji DIWA

Rysunek 4.8. Pobieranie pliku DIWA

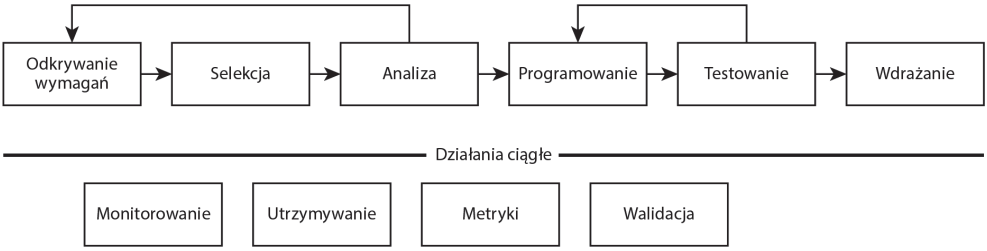


Rysunek 4.9. Rozszerzony widok dokumentu

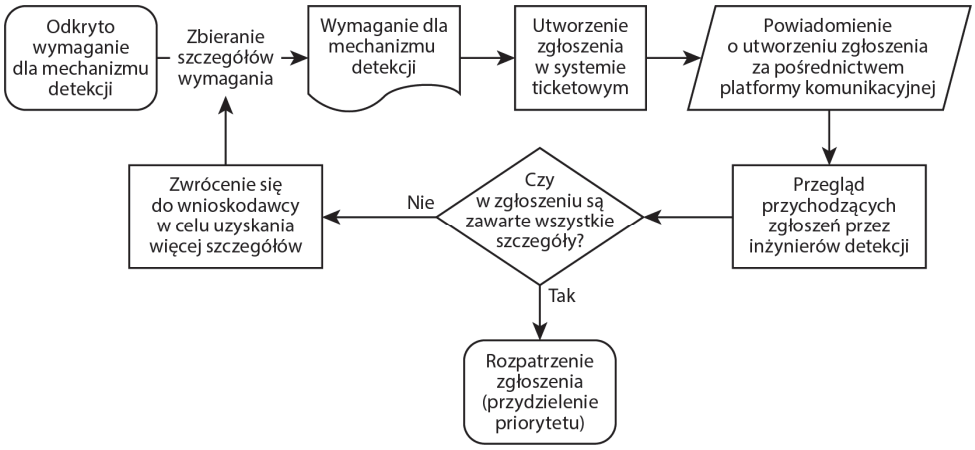


Rysunek 4.10. Alert o uzyskaniu dostępu do poufnego dokumentu

Rozdział 5. Analiza wymagań dla mechanizmów detekcji



Rysunek 5.1. Cykl życia mechanizmu detekcji



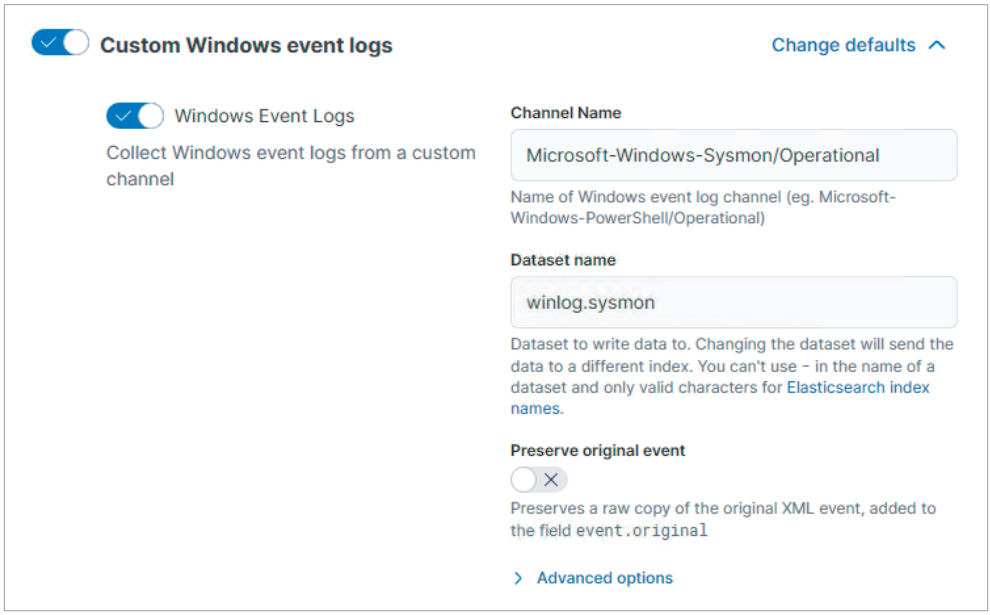
Rysunek 5.2. Przebieg fazy odkrywania wymagań

$$\text{Dotkliwość zagrożenia} + \text{Dopasowanie mechanizmu detekcji zagrożenia do organizacji} + \text{Pokrycie wymagań} + (\text{Trafność} + \text{Częstość występowania})$$

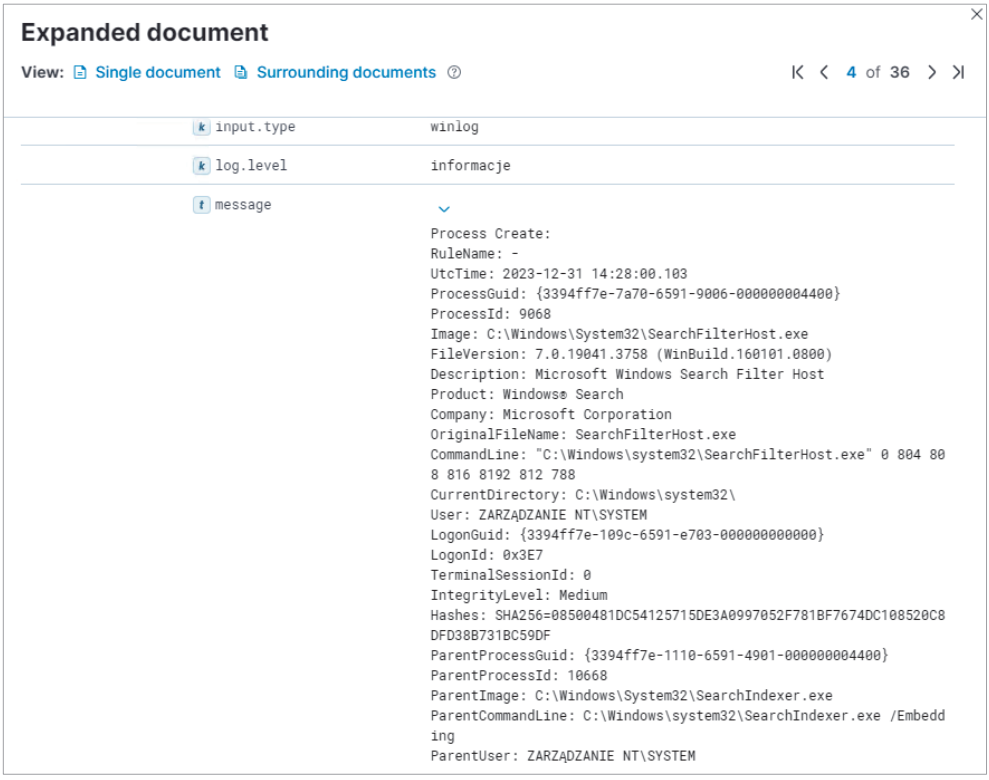
Aktywne eksploity

Rysunek 5.3. Wzór na obliczanie priorytetu

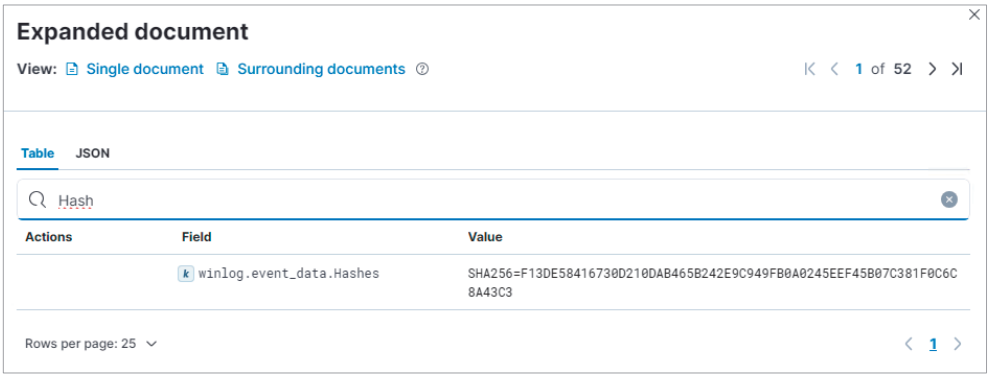
Rozdział 6. Tworzenie mechanizmów detekcji przy użyciu wskaźników naruszeń zabezpieczeń



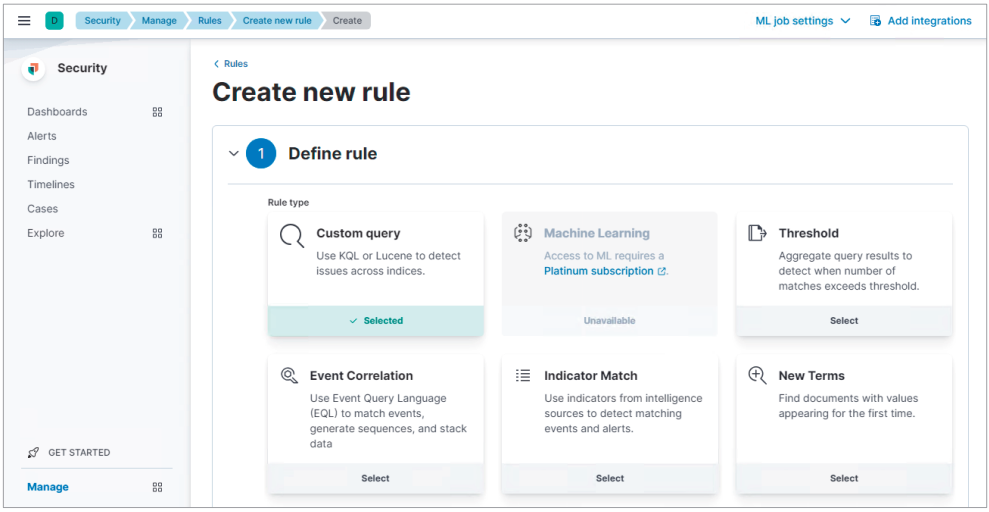
Rysunek 6.1. Dodawanie integracji Sysmon



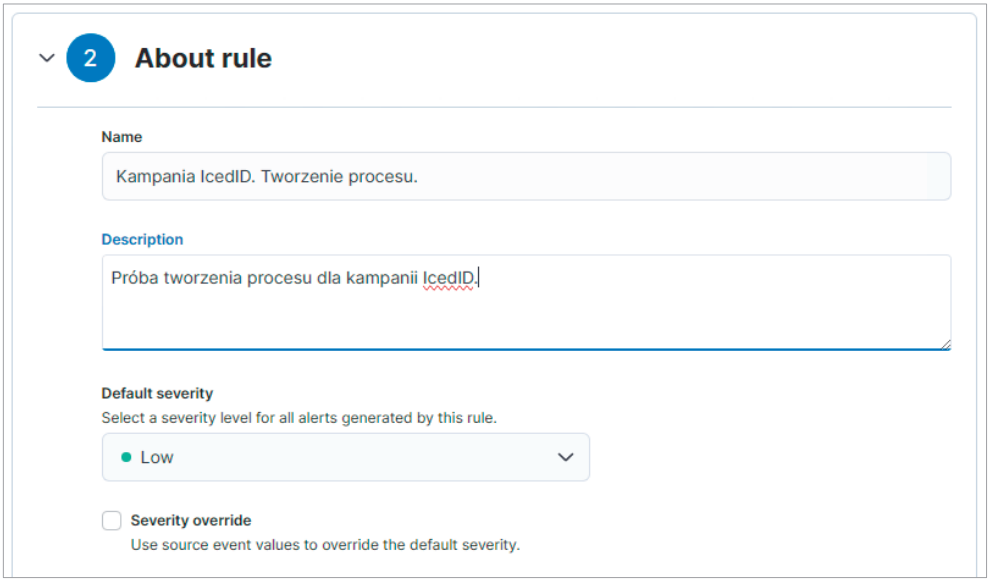
Rysunek 6.2. Przykładowe zdarzenie o identyfikatorze 1



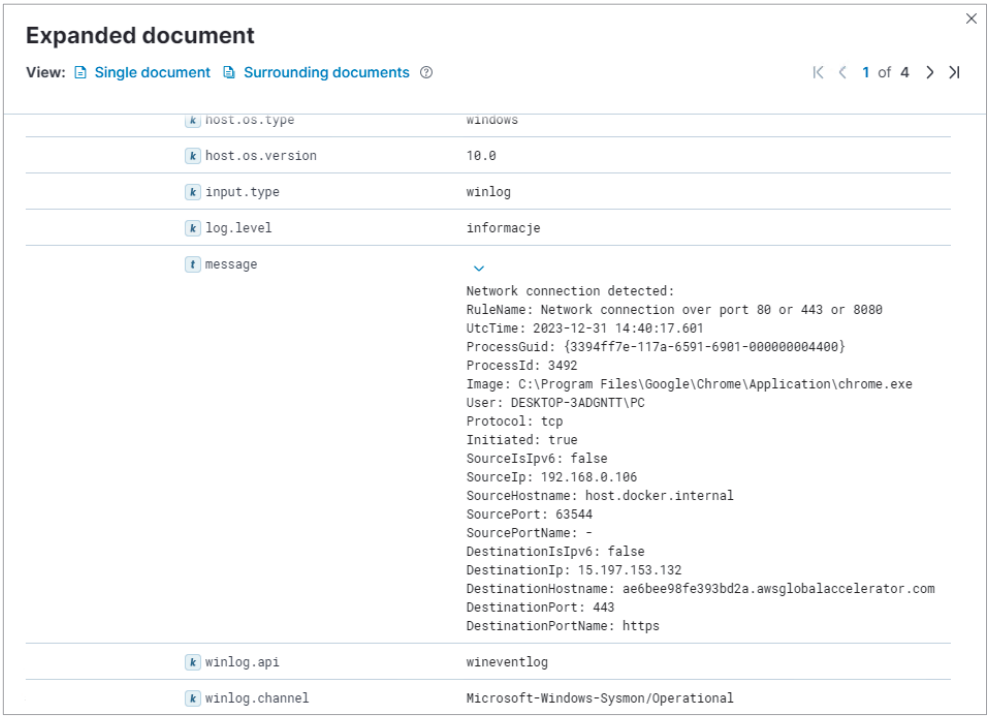
Rysunek 6.3. Pola zawierające skróty dla zdarzenia o identyfikatorze 1



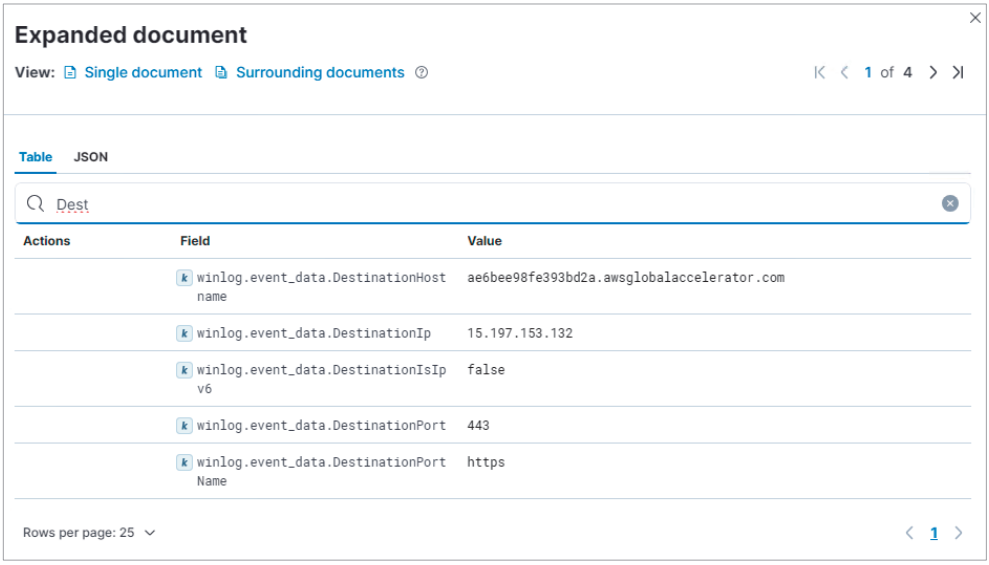
Rysunek 6.4. Konfiguracja nowej reguły



Rysunek 6.5. Konfiguracja reguły About rule

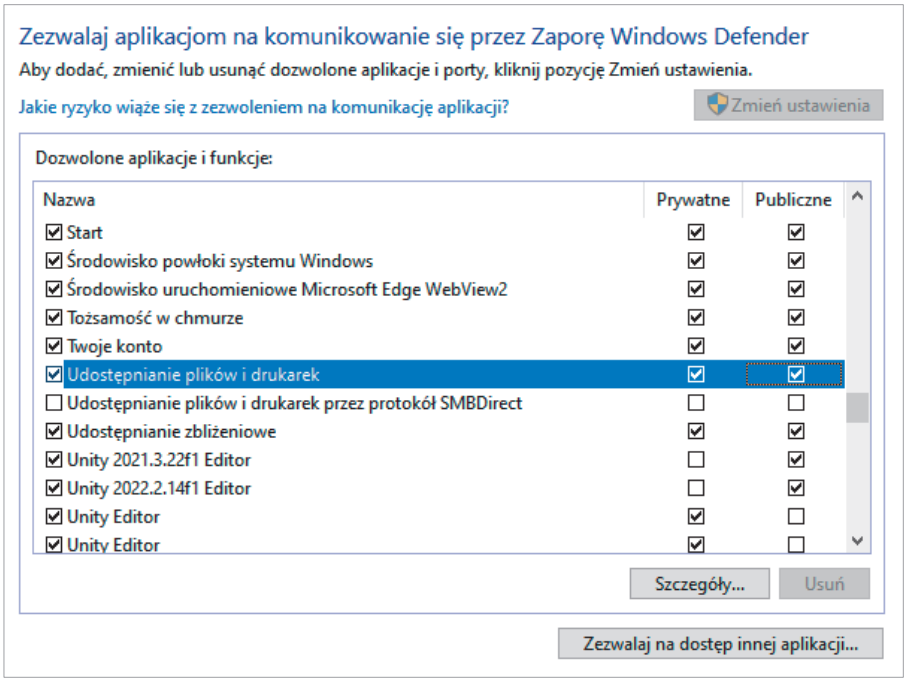


Rysunek 6.6. Przykładowe zdarzenie o identyfikatorze 3

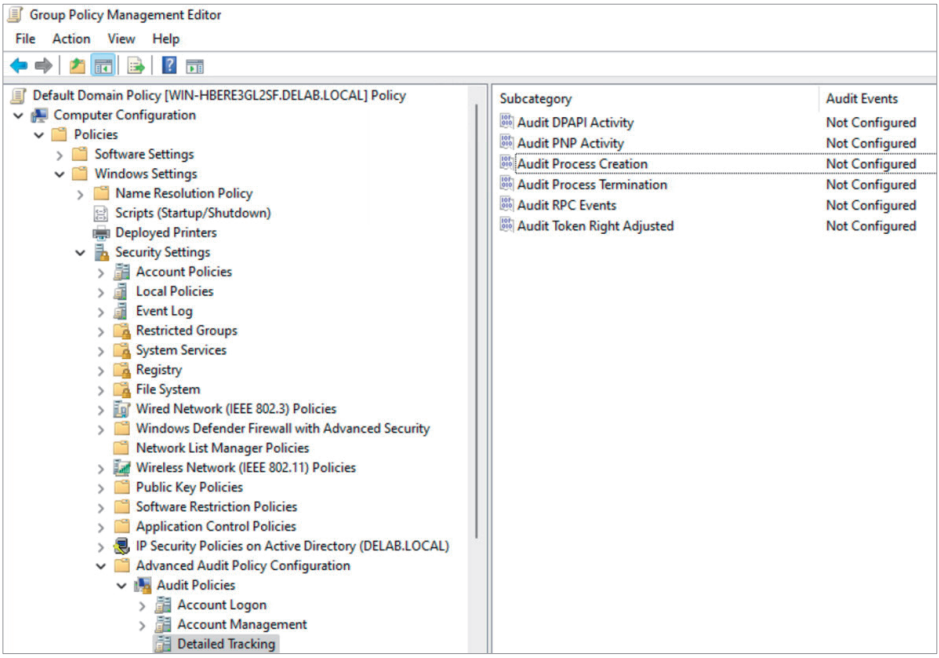


Rysunek 6.7. Pola opisujące docelowy ruch dla zdarzenia o identyfikatorze 3

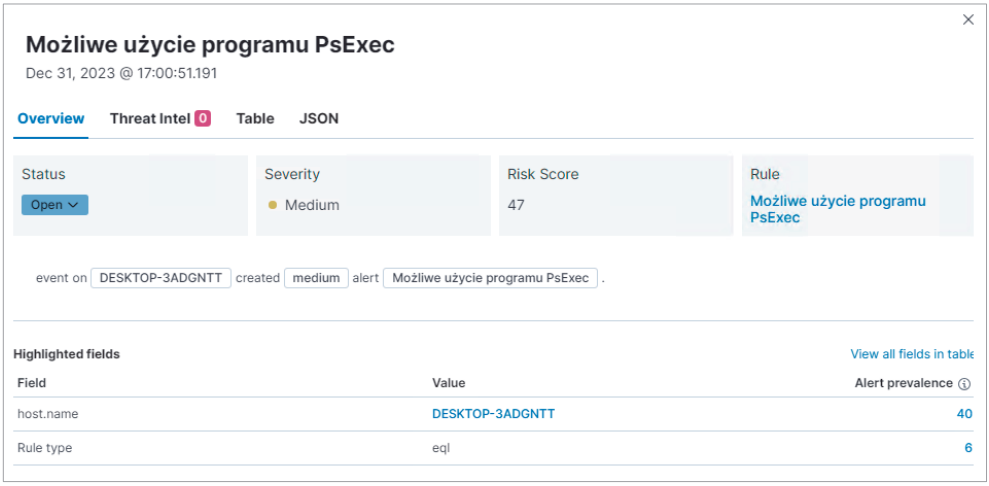
Rozdział 7. Opracowywanie mechanizmów detekcji opartych na wskaźnikach behawioralnych



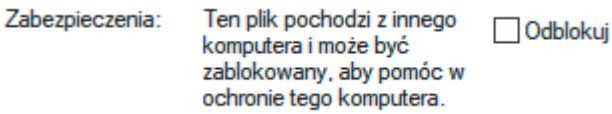
Rysunek 7.1. Zezwalanie na udostępnianie plików i drukarek



Rysunek 7.2. Edytor Group Policy Management



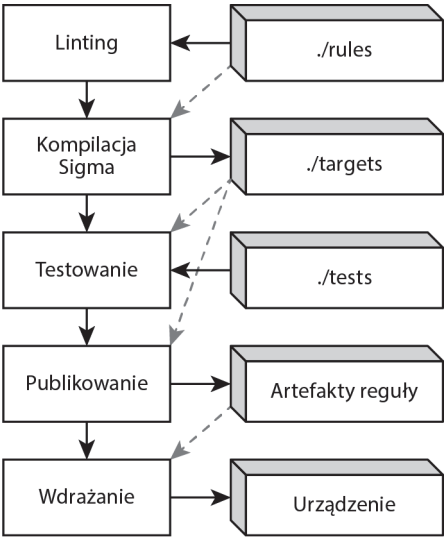
Rysunek 7.3. Wyzwolony alert możliwego użycia PsExec



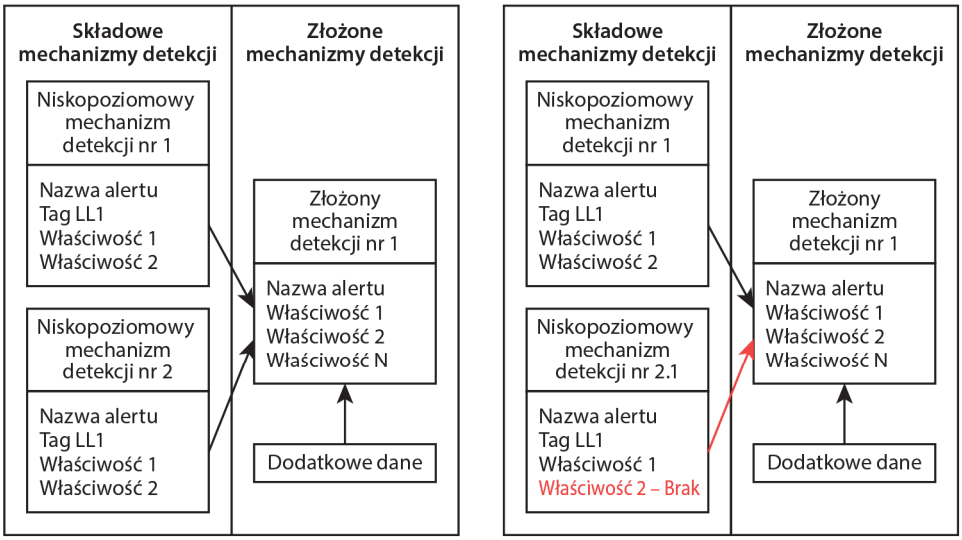
Rysunek 7.4. Znacznik sieciowy w systemie Windows

Rozdział 8. Tworzenie dokumentacji i potoki mechanizmów detekcji

Rysunek 8.1. Układ repozytorium mechanizmów detekcji



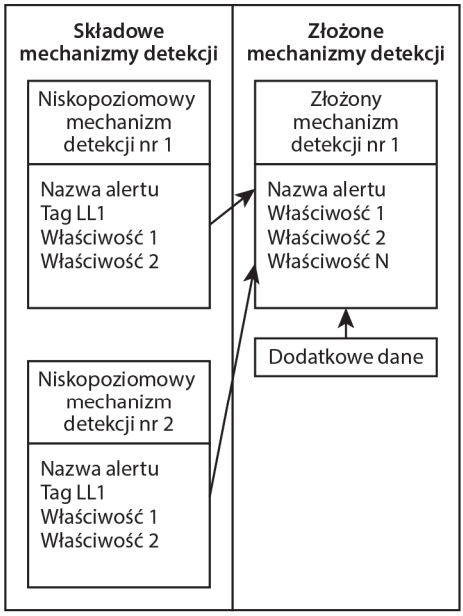
Rysunek 8.2. Potok ciągłego wdrażania dla reguł Sigma



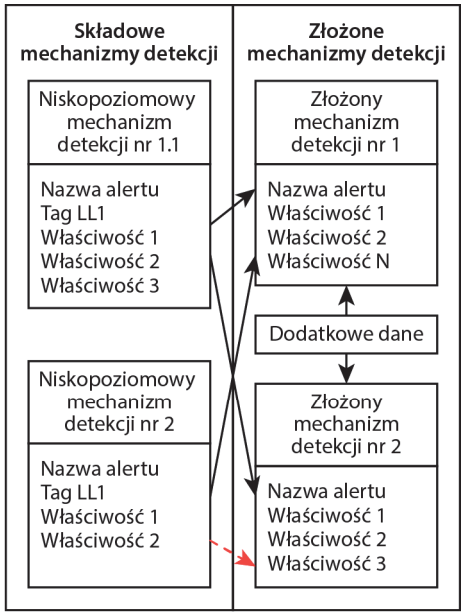
Złożony mechanizm detekcji nr 1 jest generowany w oparciu o alerty oznaczone znacznikiem LL1, które mogły zostać wygenerowane za pomocą niskopoziomowych mechanizmów detekcji nr 1 i nr 2 i wymagają do uaktywnienia właściwości 1 i 2

Niskopoziomowy mechanizm detekcji nr 2 został zaktualizowany do wersji 2.1, a właściwość 2 nie jest już uwzględniona w alercie lub jest nieprawidłowo sformatowana. Złożony mechanizm detekcji nr 1 nadal się uaktywnia, ale tylko dla niskopoziomowego mechanizmu detekcji nr 1. Zespół SOC nadal widzi alerty złożonego mechanizmu detekcji nr 1, nie zdając sobie sprawy, że brakuje niektórych alertów

Rysunek 8.3. Składowe mechanizmy detekcji i mechanizmy złożone — przykład nr 1

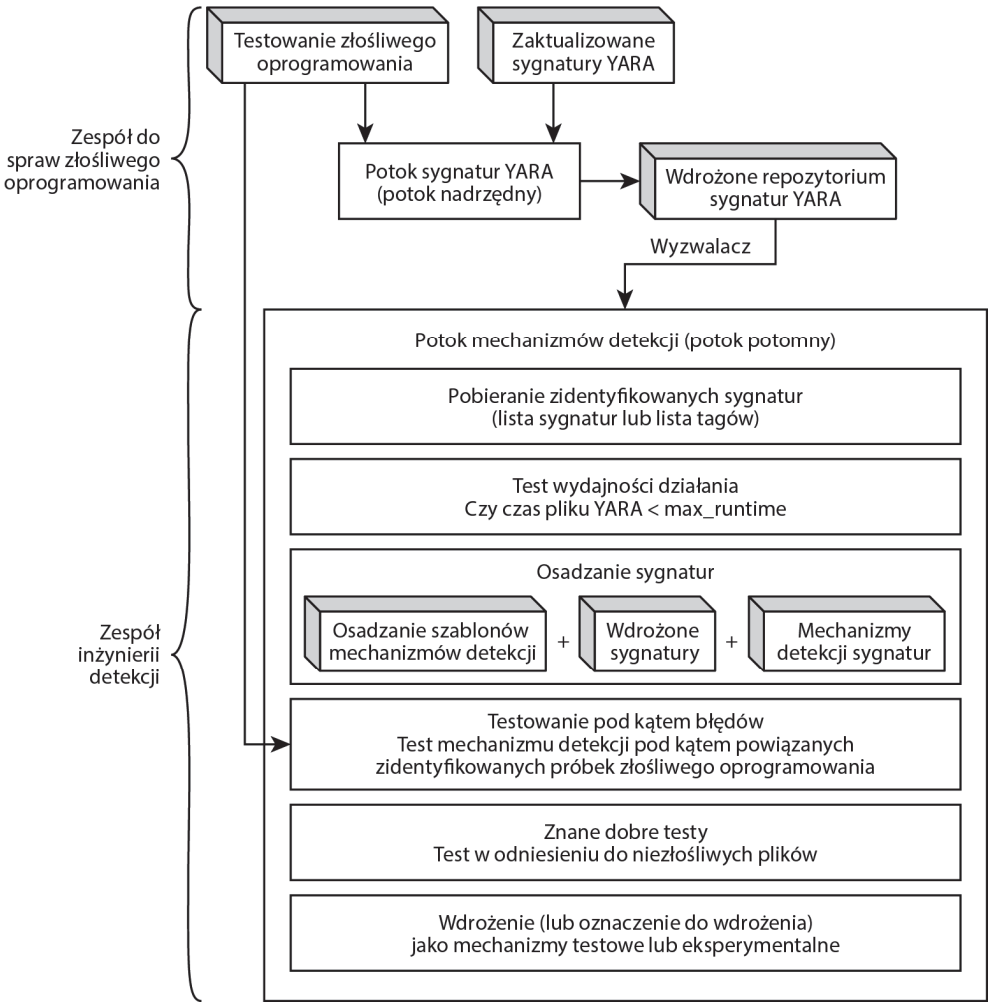


Złożony mechanizm detekcji nr 1 jest generowany w oparciu o alerty oznaczone znacznikiem LL1, które mogły zostać wygenerowane za pomocą niskopoziomowych mechanizmów detekcji nr 1 i nr 2 i wymagają do uaktywnienia właściwości 1 i 2

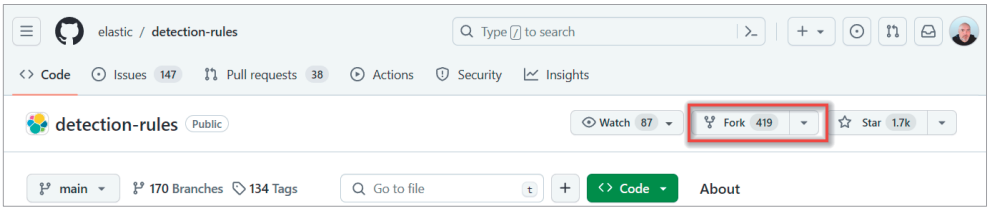


Niskopoziomowy mechanizm detekcji nr 1 został zaktualizowany do wersji 1.1 i dodano właściwość 3 w celu obsługi złożonego mechanizmu detekcji nr 2, ale analityk zapomniał zaktualizować również niskopoziomowy mechanizm detekcji nr 2, przez co nie wszystkie niskopoziomowe mechanizmy detekcji dostarczają informacji pozwalających na poprawne uaktywnienie złożonego mechanizmu detekcji nr 2

Rysunek 8.4. Składowe mechanizmy detekcji i mechanizmy złożone — przykład nr 2



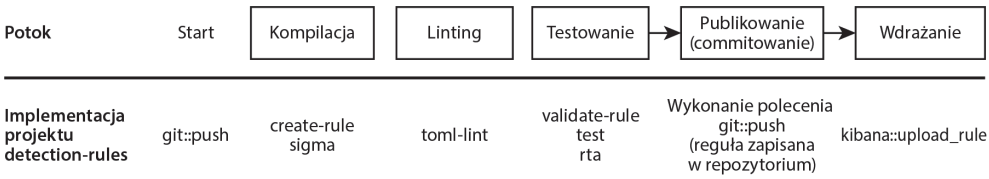
Rysunek 8.5. Przykładowy potok mechanizmu detekcji jako kodu z osadzaniem sygnatur YARA



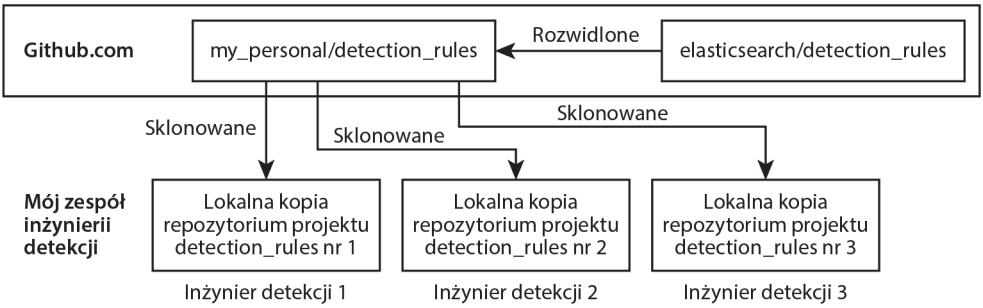
Rysunek 8.6. Tworzenie forka repozytorium detection-rules

```
(env) D:\gitrepo\detection_rules>python -m detection_rules --help

 ____       _            _   _             ____        _
|  _ \  ___| |_ ___  ___| |_(_) ___  _ __ |  _ \ _   _| | ___  ___
| | | |/ _ \ __/ _ \/ __| __| |/ _ \| '_ \| |_) | | | | |/ _ \/ __|
| |_| |  __/ ||  __/ (__| |_| | (_) | | | |  _ <| |_| | |  __/\__ \
|____/ \___|\__\___|\___|\__|_|\___/|_| |_|_| \_\\__,_|_|\___||___/

Usage: detection_rules [OPTIONS] COMMAND [ARGS]...

  Commands for detection-rules repository.

Options:
  -D, --debug / -N, --no-debug  Print full exception stacktrace on errors
  -h, --help                    Show this message and exit.

Commands:
  build-limited-rules     Import rules from json, toml, or Kibana...
  build-threat-map-entry  Build a threat map entry.
  create-rule             Create a detection rule.
  dev                     Commands related to the Elastic Stack rules...
  es                      Commands for integrating with Elasticsearch.
  export-rules            Export rule(s) into an importable ndjson file.
  generate-rules-index    Generate enriched indexes of rules, based on a...
  import-rules            Import rules from json, toml, yaml, or Kibana...
  kibana                  Commands for integrating with Kibana.
  mass-update             Update multiple rules based on eql results.
  normalize-data          Normalize Elasticsearch data timestamps and sort.
  rta                     Commands related to Red Team Automation (RTA)...
  rule-search             Use KQL or EQL to find matching rules.
  test                    Run unit tests over all of the rules.
  toml-lint               Cleanup files with some simple toml formatting.
  typosquat               Commands for generating typosquat detections.
  validate-all            Check if all rules validates against a schema.
  validate-rule           Check if a rule staged in rules dir validates...
  view-rule               View an internal rule or specified rule file.

(env) D:\gitrepo\detection_rules>
```

Rysunek 8.7. Ekran pomocy projektu detection-rules



Rysunek 8.8. Potok z implementacją projektu detection-rules



Rysunek 8.9. Potok z implementacją projektu detection-rules

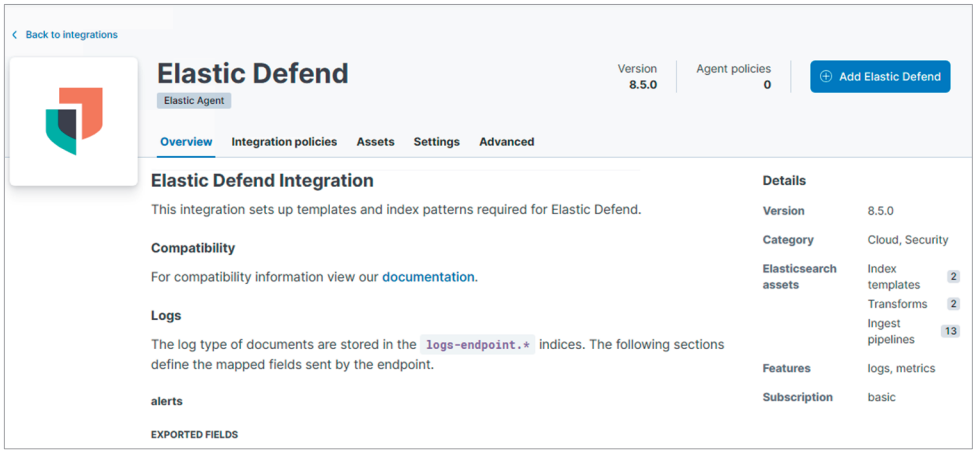
```
eql.errors.EqlSchemaError: Error at line:1,column:89
Field not recognized
any where event.code=="1" and winlog.channel=="Microsoft-Windows-VHDMP-Operational" and winlog.event_data.VhdFileName like~ ("*Temp*","*.zip*","*.iso")
                                                                                        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
stack: 8.9.0, beats: 8.9.0,ecs: 8.9.0, endgame: 8.4.0
```

Rysunek 8.10. Błąd EqlSchemaError

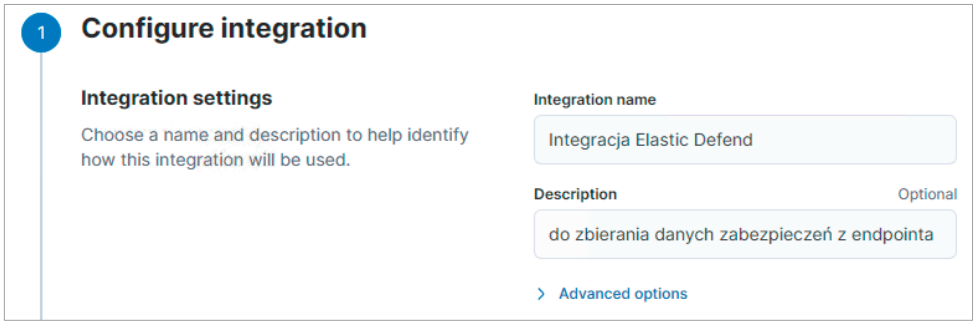
```
(env) D:\gitrepo\detection_rules>python -m detection_rules import-rules ./rules/windows/defense_evasion_iso_mounted_from_zip_temp_directory.toml
DETECTION RULES
[+] Building rule for D:\gitrepo\detection_rules\rules\obraz_iso_zamontowany_z_tymczasowego_folderu_zip.toml
```

Rysunek 8.11. Wynik działania polecenia import-rules

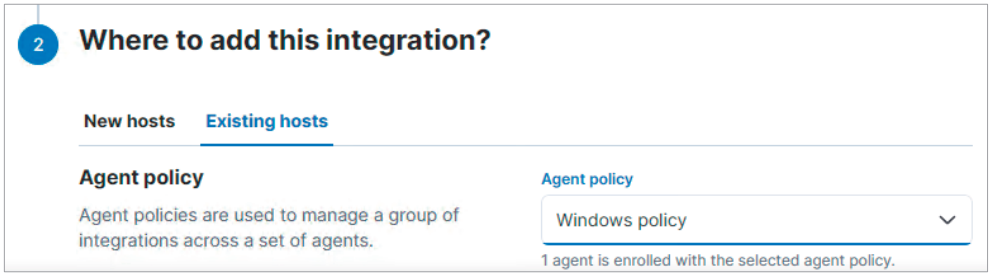
Rozdział 9. Walidacja mechanizmów detekcji



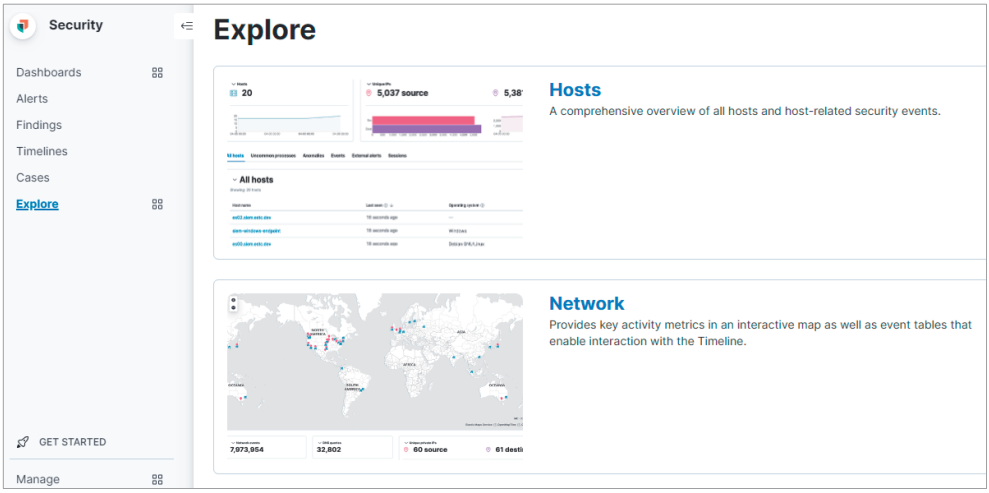
Rysunek 9.1. Integracja Elastic Defend



Rysunek 9.2. Krok nr 1 kreatora dodawania integracji

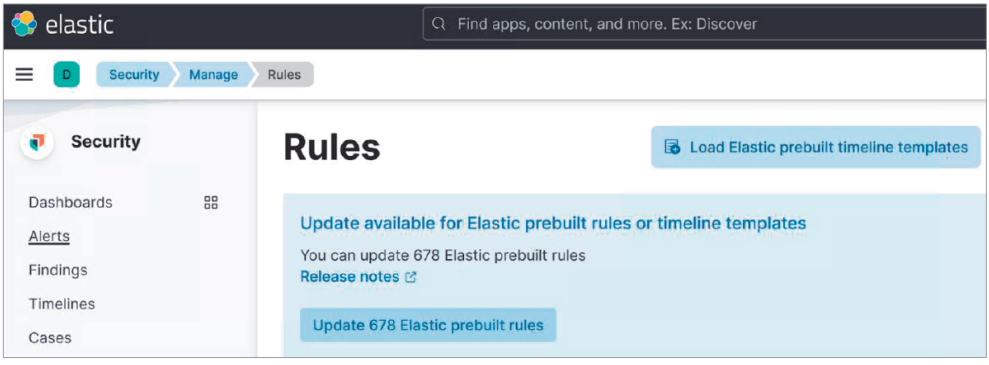


Rysunek 9.3. Krok nr 2 kreatora dodawania integracji

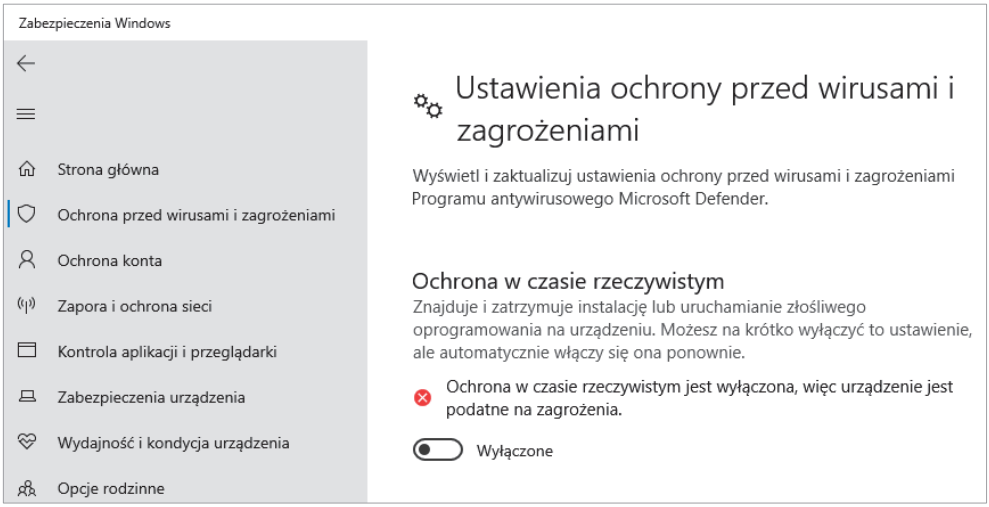


Rysunek 9.4. Aplikacja Elastic Security

Rysunek 9.5. Widok Events w aplikacji Elastic Security



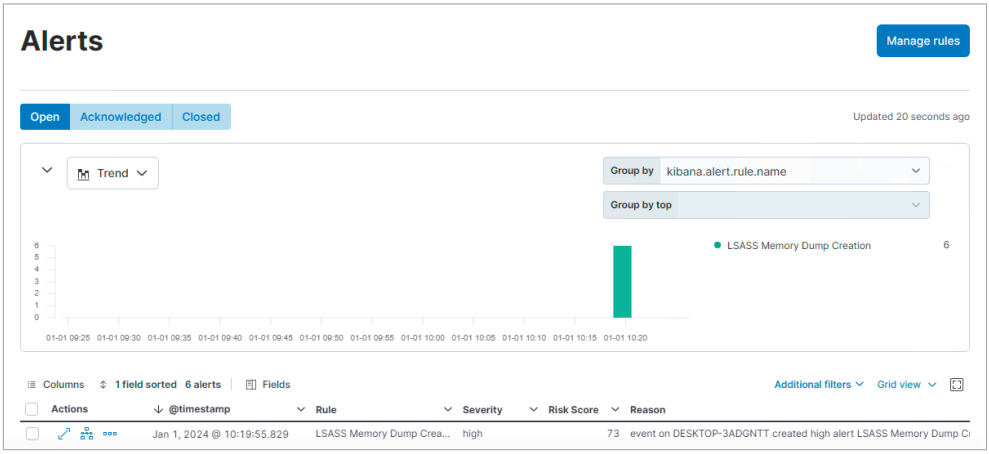
Rysunek 9.6. Zarządzanie regułami Kibana Security



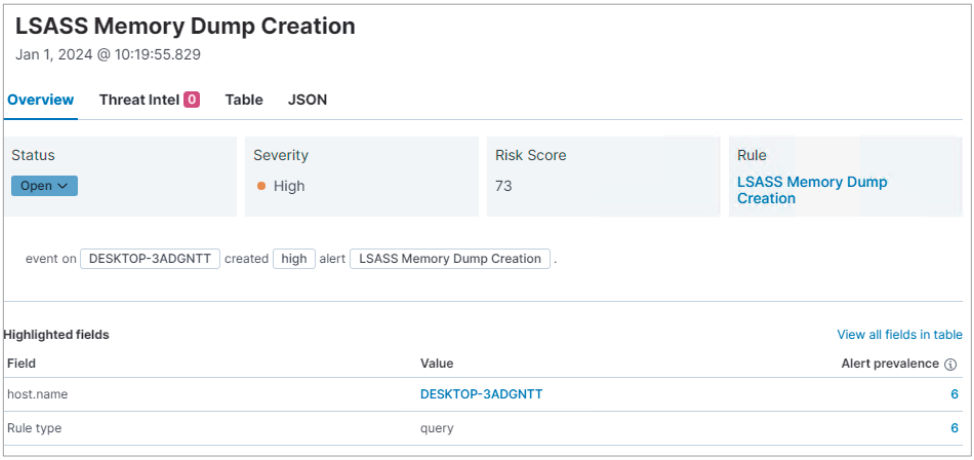
Rysunek 9.7. Wyłączanie ochrony w czasie rzeczywistym w usłudze Windows Defender

```
PS C:\procdump> ./procdump.exe -accepteula -ma lsass.exe c:\windows\temp\lsass_dump.dmp

ProcDump v11.0 - Sysinternals process dump utility
Copyright (C) 2009-2022 Mark Russinovich and Andrew Richards
Sysinternals - www.sysinternals.com

[09:00:53] Dump 1 initiated: c:\windows\temp\lsass_dump.dmp
[09:01:00] Dump 1 writing: Estimated dump file size is 61 MB.
[09:01:01] Dump 1 complete: 62 MB written in 7.6 seconds
[09:01:01] Dump count reached.

PS C:\procdump>
```

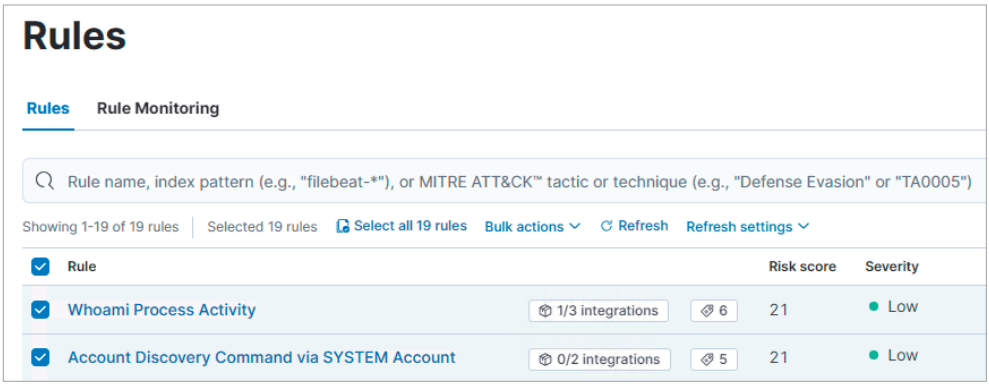
Rysunek 9.8. Uruchomienie testu Atomic Red Team



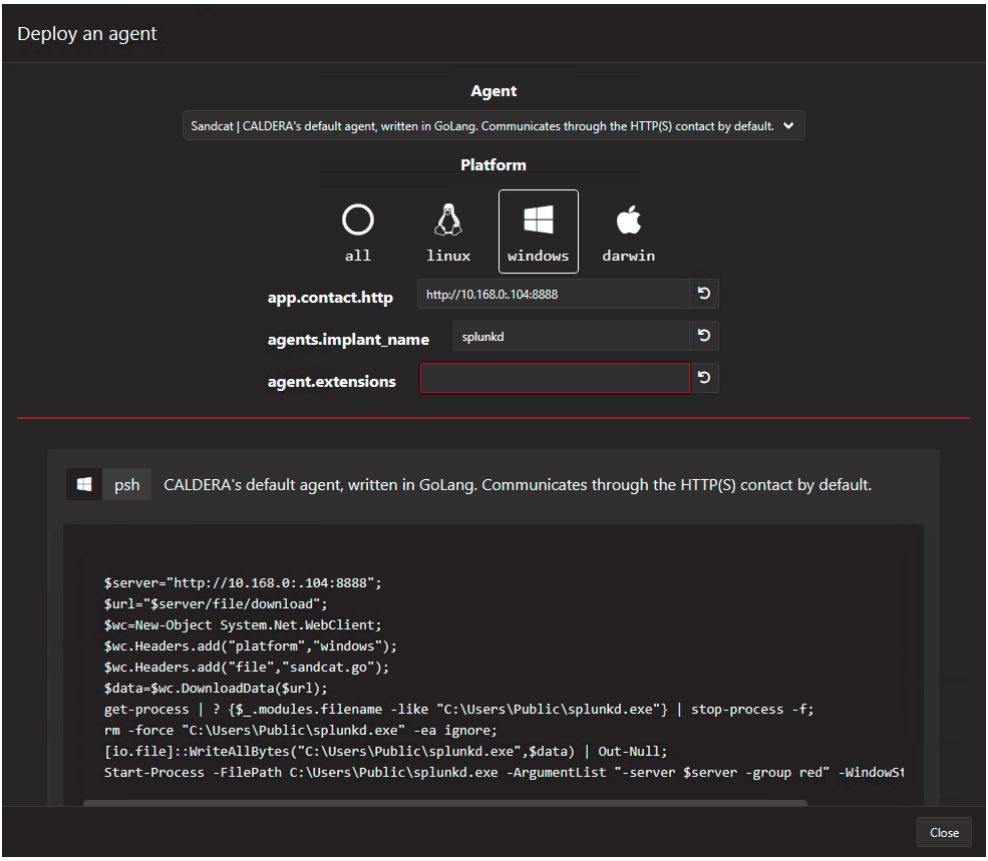
Rysunek 9.9. Alerty w aplikacji Elastic Security



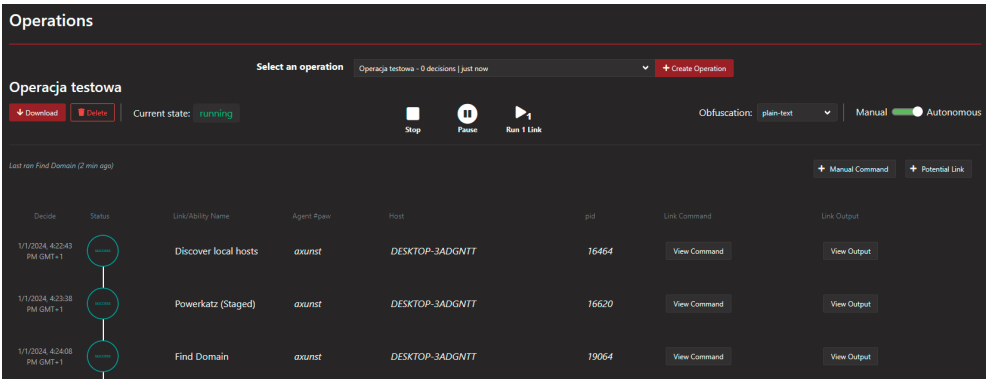
Rysunek 9.10. Alarm tworzenia zrzutu pamięci LSASS



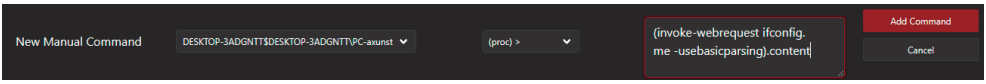
Rysunek 9.11. Reguły Windows i Discovery



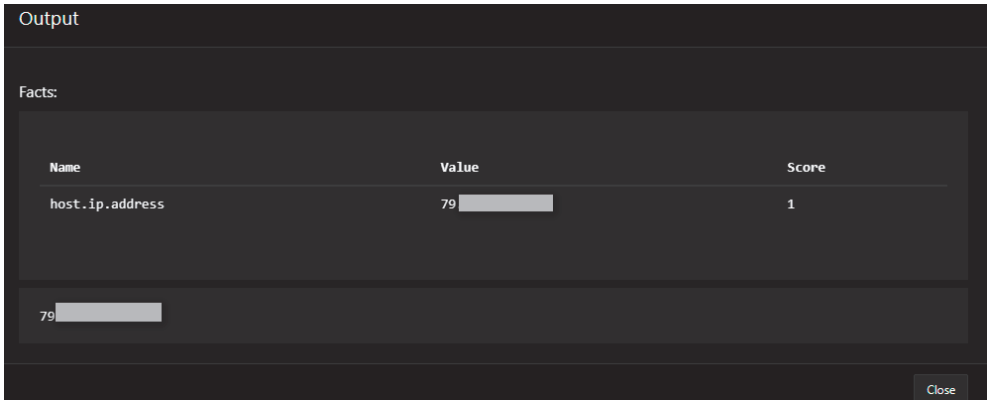
Rysunek 9.12. Instalacja agenta CALDERA



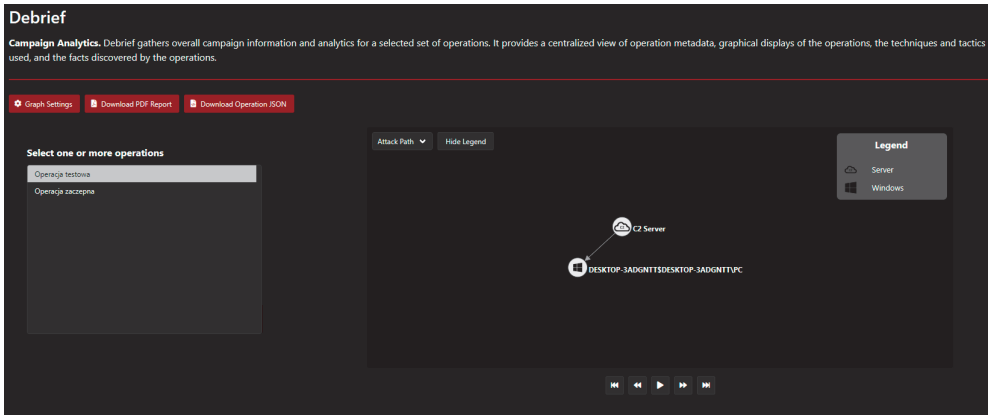
Rysunek 9.13. Operacje w systemie CALDERA



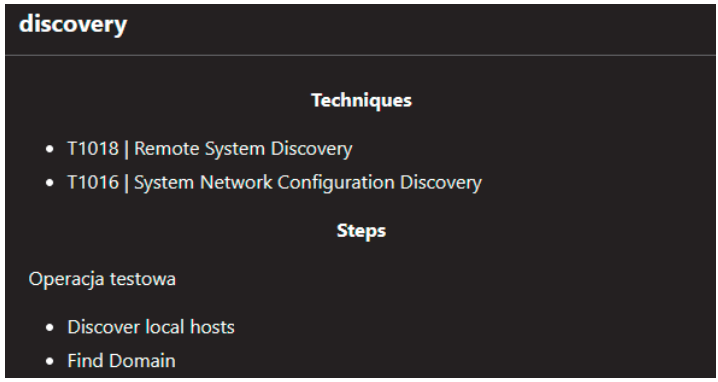
Rysunek 9.14. Ręczne polecenia w systemie CALDERA



Rysunek 9.15. Wynik działania ręcznego polecenia w systemie CALDERA



Rysunek 9.16. Wtyczka Debrief systemu CALDERA



Rysunek 9.17. Techniki wykrywania stosowane w operacji CALDERA

OPERATIONS DEBRIEF

TACTICS AND TECHNIQUES

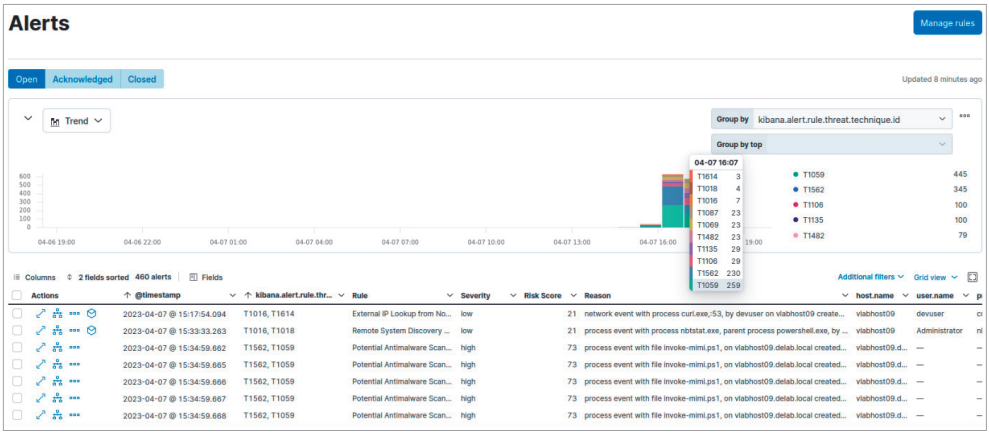
Tactics	Techniques	Abilities
Auto-generated	auto-generated: auto-generated	Operacja testowa Manual Command
Credential-access	T1003.001: OS Credential Dumping: LSASS Memory	Operacja testowa Powerkatz (Staged)
Discovery	T1018: Remote System Discovery T1016: System Network Configuration Discovery	Operacja testowa Discover local hosts Find Domain

STEPS IN OPERATION OPERACJA TESTOWA

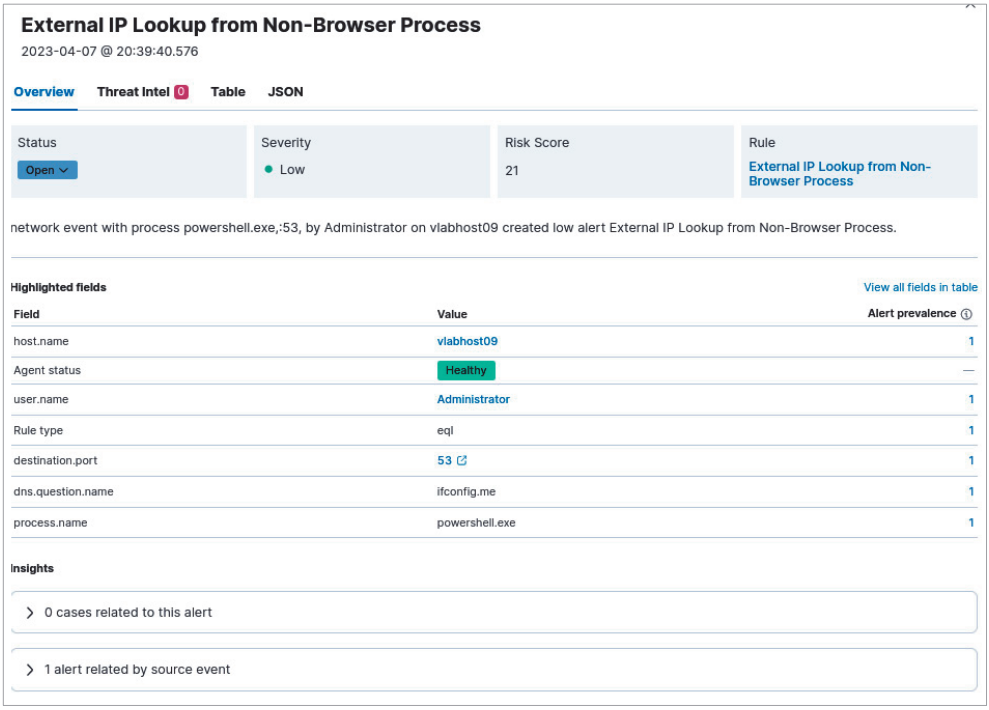
The table below shows detailed information about the steps taken in an operation and whether the command run discovered any facts.

Time	Status	Agent	Name	Command	Facts
2024-01-01 T15:23:35Z	success	axunst	Discover local hosts	Import-Module .\powerview.ps1;Get-DomainComputer	No
2024-01-01 T15:24:07Z	success	axunst	Powerkatz (Staged)	Import-Module .\invoke-mimi.ps1;Invoke-Mimikatz -DumpCreds	No
2024-01-01 T15:25:05Z	success	axunst	Find Domain	nbtstat -n	No
2024-01-01 T15:30:17Z	failure	axunst	Manual Command	(invoke-webrequest ifconfig.me -usebasicparsing).content	No
2024-01-01 T15:32:18Z	success	axunst	Manual Command	(invoke-webrequest ifconfig.me -usebasicparsing).content	Yes

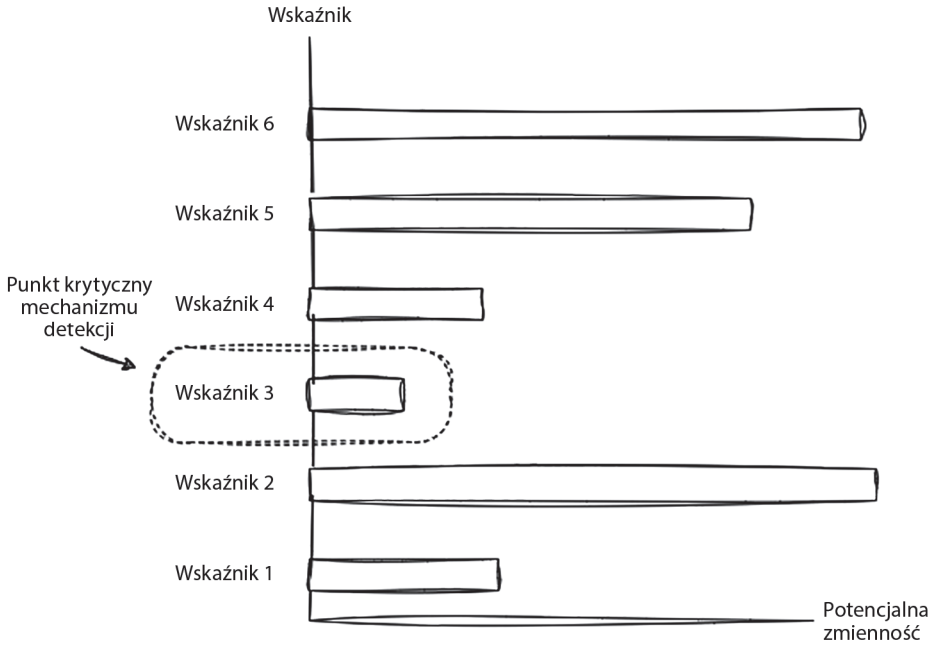
Rysunek 9.18. Raport PDF z podsumowania operacji CALDERA



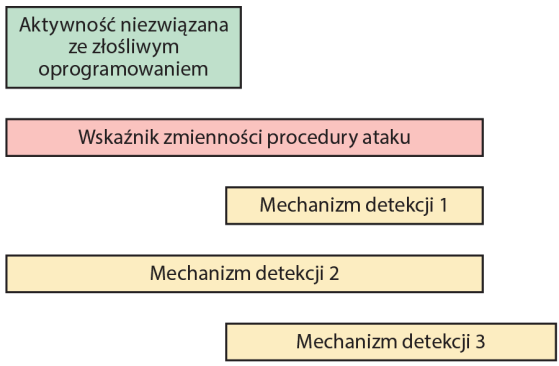
Rysunek 9.19. Zmodyfikowany widok Alerts w systemie Elastic Security



Rysunek 9.20. Alert wyszukiwania zewnętrznego adresu IP z procesu spoza przeglądarki



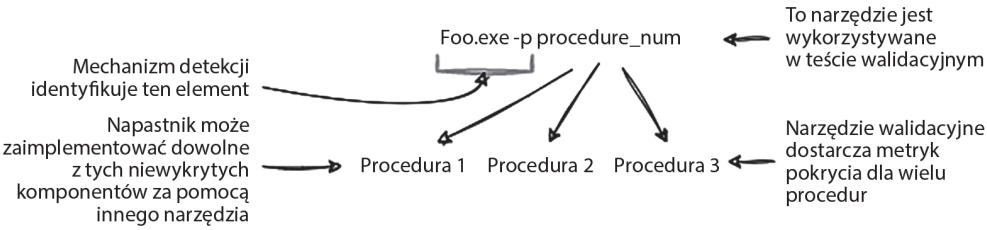
Rysunek 9.21. Wskaźniki i potencjalna zmienność



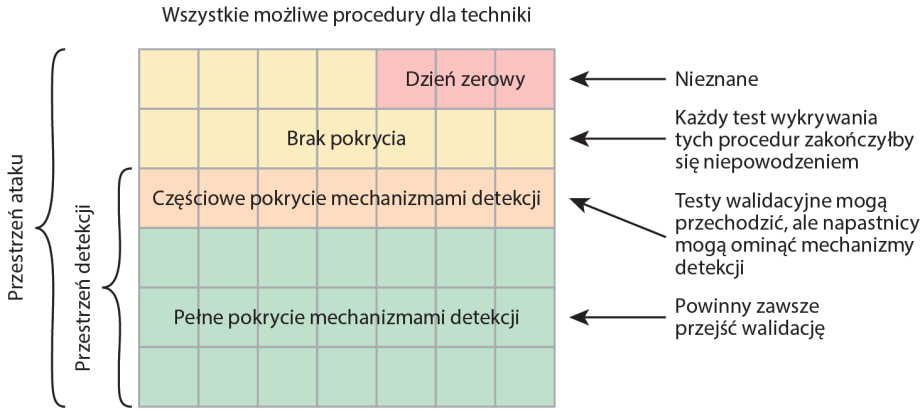
Jednowymiarowy widok przestrzeni ataków i powiązanych z nimi mechanizmów detekcji. Ten widok dodaje wymiar z każdym używanym wskaźnikiem jako warunek „i" w ramach mechanizmu detekcji

Mechanizm detekcji 1: brak fałszywych alarmów, ale także brak poprawnych wyników dodatnich
Mechanizm detekcji 2: uwzględnia fałszywe alarmy, ale nie pomija poprawnych wyników dodatnich
Mechanizm detekcji 3: obejmuje fałszywe alarmy niemożliwe do uzyskania w wyniku działania przeciwnika i obejmuje tylko niektóre wyniki poprawnie dodatnie

Rysunek 9.22. Przestrzeń mechanizmu detekcji

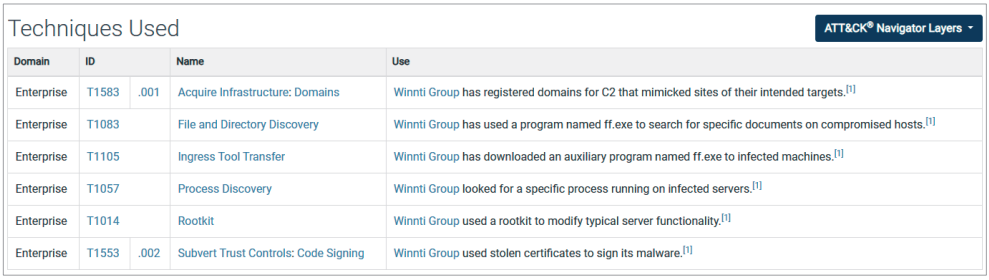


Rysunek 9.23. Narzędzia walidacyjne a procedury



Rysunek 9.24. Przestrzenie ataku i detekcji wielu procedur

Rozdział 10. Wykorzystanie wiedzy o zagrożeniach

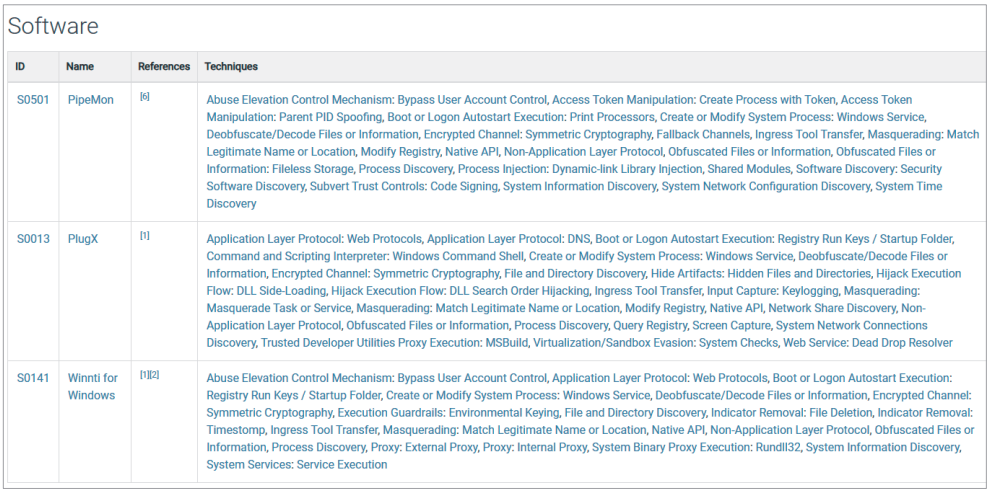


Techniques Used

ATT&CK® Navigator Layers

Domain	ID		Name	Use
Enterprise	T1583	.001	Acquire Infrastructure: Domains	Winnti Group has registered domains for C2 that mimicked sites of their intended targets.[1]
Enterprise	T1083		File and Directory Discovery	Winnti Group has used a program named ff.exe to search for specific documents on compromised hosts.[1]
Enterprise	T1105		Ingress Tool Transfer	Winnti Group has downloaded an auxiliary program named ff.exe to infected machines.[1]
Enterprise	T1057		Process Discovery	Winnti Group looked for a specific process running on infected servers.[1]
Enterprise	T1014		Rootkit	Winnti Group used a rootkit to modify typical server functionality.[1]
Enterprise	T1553	.002	Subvert Trust Controls: Code Signing	Winnti Group used stolen certificates to sign its malware.[1]

Rysunek 10.1. Winnti Group — wykorzystywane techniki

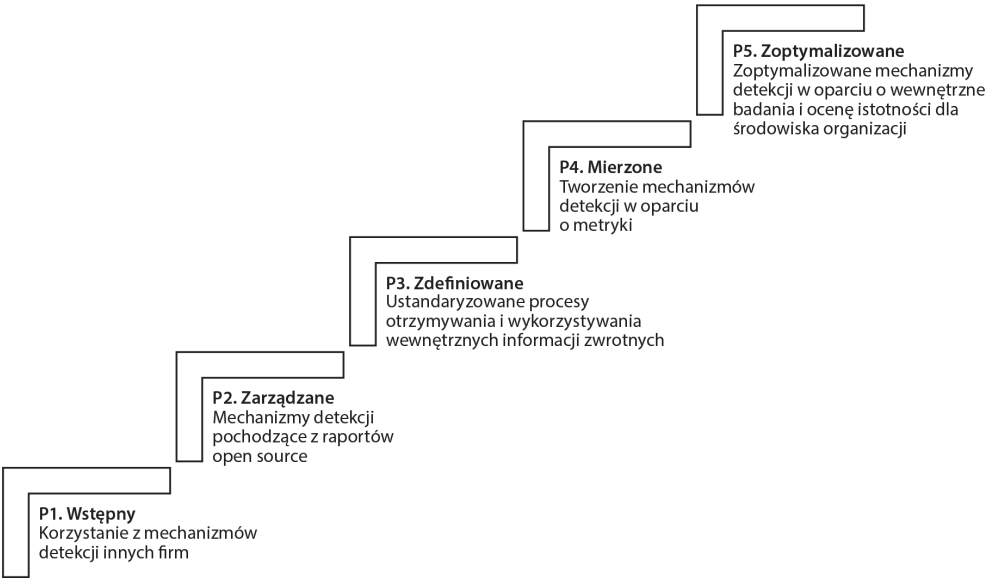


Software

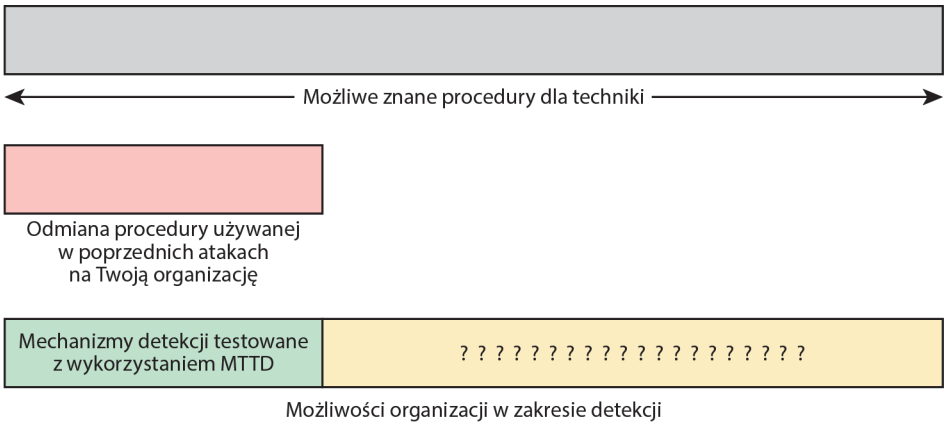
ID	Name	References	Techniques
S0501	PipeMon	[6]	Abuse Elevation Control Mechanism: Bypass User Account Control, Access Token Manipulation: Create Process with Token, Access Token Manipulation: Parent PID Spoofing, Boot or Logon Autostart Execution: Print Processors, Create or Modify System Process: Windows Service, Deobfuscate/Decode Files or Information, Encrypted Channel: Symmetric Cryptography, Fallback Channels, Ingress Tool Transfer, Masquerading: Match Legitimate Name or Location, Modify Registry, Native API, Non-Application Layer Protocol, Obfuscated Files or Information, Obfuscated Files or Information: Fileless Storage, Process Discovery, Process Injection: Dynamic-link Library Injection, Shared Modules, Software Discovery: Security Software Discovery, Subvert Trust Controls: Code Signing, System Information Discovery, System Network Configuration Discovery, System Time Discovery
S0013	PlugX	[1]	Application Layer Protocol: Web Protocols, Application Layer Protocol: DNS, Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder, Command and Scripting Interpreter: Windows Command Shell, Create or Modify System Process: Windows Service, Deobfuscate/Decode Files or Information, Encrypted Channel: Symmetric Cryptography, File and Directory Discovery, Hide Artifacts: Hidden Files and Directories, Hijack Execution Flow: DLL Side-Loading, Hijack Execution Flow: DLL Search Order Hijacking, Ingress Tool Transfer, Input Capture: Keylogging, Masquerading: Masquerade Task or Service, Masquerading: Match Legitimate Name or Location, Modify Registry, Native API, Network Share Discovery, Non-Application Layer Protocol, Obfuscated Files or Information, Process Discovery, Query Registry, Screen Capture, System Network Connections Discovery, Trusted Developer Utilities Proxy Execution: MSBuild, Virtualization/Sandbox Evasion: System Checks, Web Service: Dead Drop Resolver
S0141	Winnti for Windows	[1][2]	Abuse Elevation Control Mechanism: Bypass User Account Control, Application Layer Protocol: Web Protocols, Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder, Create or Modify System Process: Windows Service, Deobfuscate/Decode Files or Information, Encrypted Channel: Symmetric Cryptography, Execution Guardrails: Environmental Keying, File and Directory Discovery, Indicator Removal: File Deletion, Indicator Removal: Timestomp, Ingress Tool Transfer, Masquerading: Match Legitimate Name or Location, Native API, Non-Application Layer Protocol, Obfuscated Files or Information, Process Discovery, Proxy: External Proxy, Proxy: Internal Proxy, System Binary Proxy Execution: Rundll32, System Information Discovery, System Services: Service Execution

Rysunek 10.2. Winnti Group — oprogramowanie

Rozdzial 11. Zarządzanie wydajnością



Rysunek 11.1. Model dojrzałości inżynierii detekcji

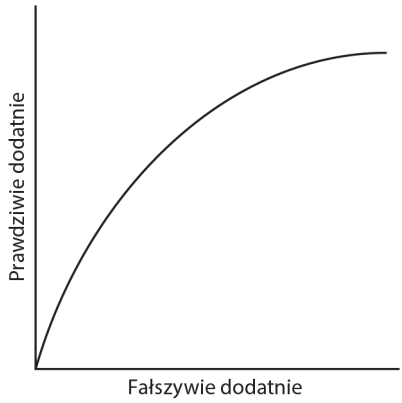


Rysunek 11.2. Testowanie mechanizmów detekcji za pomocą MTTD

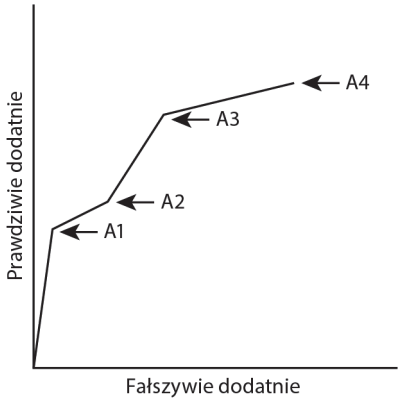
	Metryki	Niski wpływ	Średni/ wysoki	Wysoki wpływ
	Wydajność	×	×	×
	Trafność i czułość		×	×
Skuteczność	Niski poziom wierności		×	×
Skuteczność	Automatyczna walidacja		×	×
Skuteczność	Wysoki poziom wierności			×

Rysunek 11.3. Metryki według poziomów ważności

Rysunek 11.4. Określanie alertów

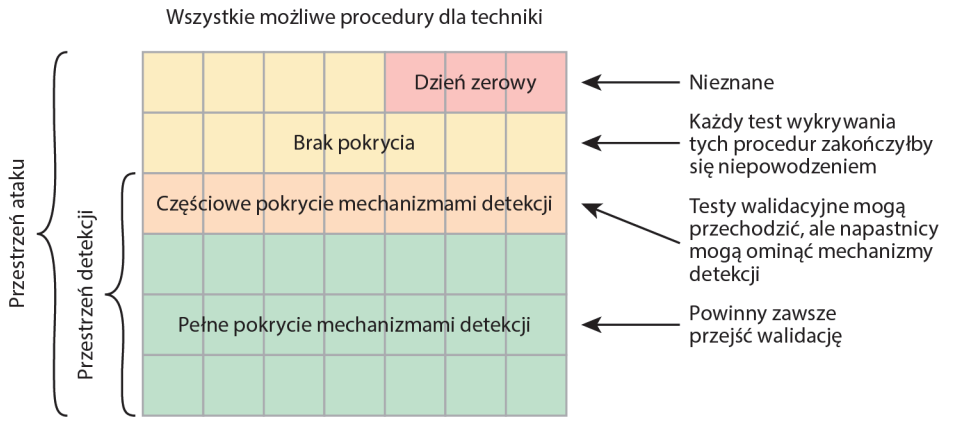


Rysunek 11.5. Standardowa krzywa ROC

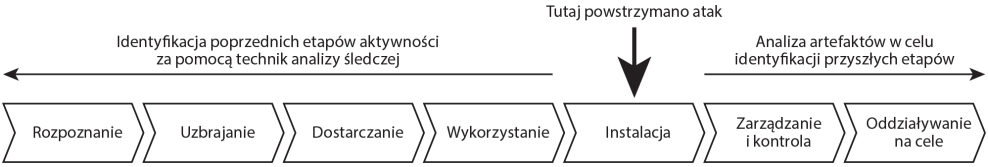


Rysunek 11.6. Ocena podejść do mechanizmów detekcji z wykorzystaniem krzywej ROC

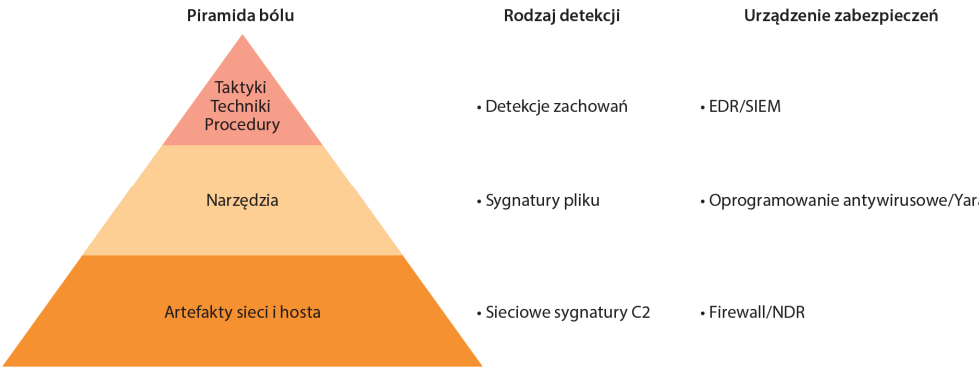
Rysunek 11.7. Przykład macierzy MITRE ATT&CK



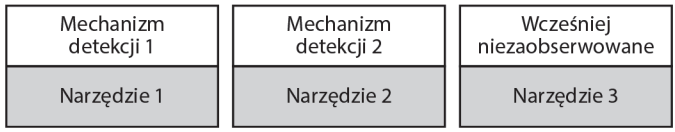
Rysunek 11. 8. Kategorie wyników walidacji mechanizmów detekcji



Rysunek 11.9. Analiza Cyber Kill Chain w celu zidentyfikowania niewykrytych działań napastnika

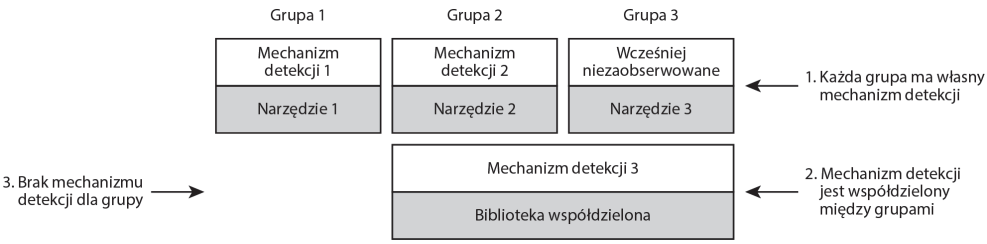


Rysunek 11.10. Piramida bólu i rodzaje detekcji

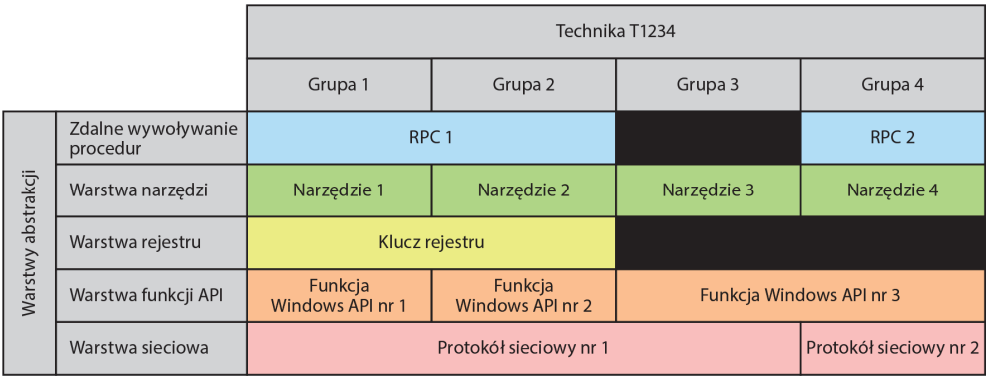


	Narzędzie 3	Narzędzie 3	Narzędzie 3
Oczekiwane wykrycia	2	2	2
Zaobserwowane wykrycia	1	1	1

Rysunek 11.11. Dryf mechanizmu detekcji dla wielu narzędzi stosowanych przez atakującego



Rysunek 11.12. Różnice w pokryciu mechanizmów detekcji



Rysunek 11.13. Mapa abstrakcji z pięcioma warstwami

		Grupy możliwości			
		Grupa 1	Grupa 2	Grupa 3	Grupa 4
Warstwy abstrakcji	Warstwa abstrakcji 1	Szczegóły artefaktu			Szczegóły artefaktu
	Warstwa abstrakcji 2	Szczegóły artefaktu	Szczegóły artefaktu	Szczegóły artefaktu	Szczegóły artefaktu
	Warstwa abstrakcji 3	Szczegóły artefaktu			
	Warstwa abstrakcji 4	Szczegóły artefaktu	Szczegóły artefaktu	Szczegóły artefaktu	
	Warstwa abstrakcji 5	Szczegóły artefaktu			Szczegóły artefaktu

Rysunek 11.14. Uogólniona mapa abstrakcji

Mapa abstrakcji ze znacznikami minimalnej liczby detekcji

		Technika T1234			
		Grupa 1	Grupa 2	Grupa 3	Grupa 4
Warstwy abstrakcji	Zdalne wywoływanie procedur	TAG: T1234 MD: 5 RPC 1			TAG: T1234 MD: 4 RPC 2
	Warstwa narzędzi	TAG: T1234 MD: 5 Narzędzie 1	TAG: T1234 MD: 5 Narzędzie 2	TAG: T1234 MD: 3 Narzędzie 3	TAG: T1234 MD: 4 Narzędzie 4
	Warstwa rejestru	TAG: T1234 MD: 5 Klucz rejestru			
	Warstwa funkcji API	TAG: T1234 MD: 5 Funkcja Win API nr 1	TAG: T1234 MD: 5 Funkcja Win API nr 2	TAG: T1234 MD: 3 Funkcja Windows API nr 3	
	Warstwa sieciowa	TAG: T1234 MD: 3 Protokół sieciowy nr 1			TAG: T1234 MD: 4 Protokół sieciowy nr 2

Rysunek 11.15. Przykład oznaczania MD

$$Minimalny\ dryf\ mechanizmu\ detekcji = Max(MD_1, MD_2, MD_n) - n$$

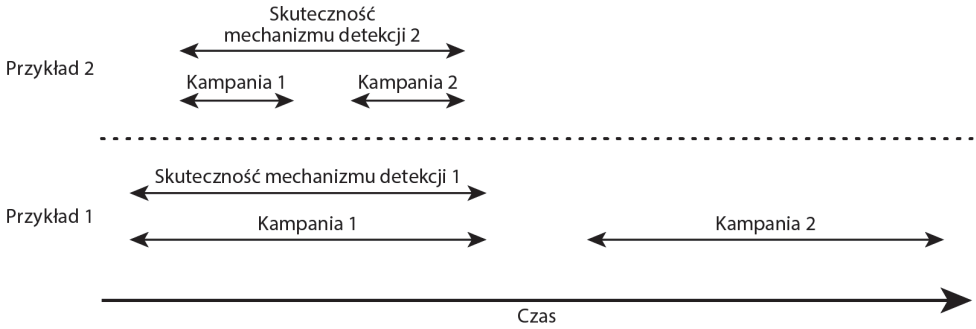
Gdzie: n = liczba alertów

Rysunek 11.16. Wzór na dryf MD

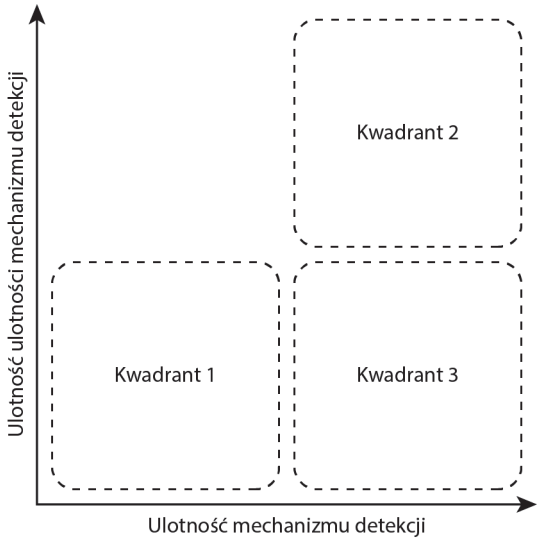
Mapa abstrakcji ze znacznikami minimalnej liczby detekcji i odpowiedzią EDR

			Technika T1234			
			Grupa 1	Grupa 2	Grupa 3	Grupa 4
EDR zatrzymuje atak, jeśli zostanie wykryty →	Warstwy abstrakcji	Zdalne wywoływanie procedur	TAG: T1234 MD: 4 RPC 1			TAG: T1234 MD: 3 RPC 2
		Warstwa narzędzi	TAG: T1234 MD: 4 Narzędzie 1	TAG: T1234 MD: 4 Narzędzie 2	TAG: T1234 MD: 2 Narzędzie 3	TAG: T1234 MD: 3 Narzędzie 4
		Warstwa rejestru	TAG: T1234 MD: 4 Klucz rejestru			
		Warstwa funkcji API	TAG: T1234 MD: 4 Funkcja Win API nr 1	TAG: T1234 MD: 4 Funkcja Win API nr 2	TAG: T1234 MD: 2 Funkcja Windows API nr 3	
		Warstwa sieciowa	TAG: T1234 MD: 3 Protokół sieciowy nr 1			TAG: T1234 MD: 4 Protokół sieciowy nr 2

Rysunek 11.17. Mapa abstrakcji z odpowiedzią EDR



Rysunek 11.18. Skuteczność mechanizmów detekcji w czasie



Rysunek 11.19. Wykres aspektów ulotności mechanizmów detekcji