

Bezpieczeństwo infrastruktury krytycznej i systemów sterowania przemysłowego IoT - Laboratorium 11 – SOC/ Wykrywanie Incydentów Part 2

Forma: praca w samodzielna / małe zespoły (2 - 4 osoby)

Wstęp teoretyczny

IDS

<https://kapitanhack.pl/co-to-jest-ids/>

<https://sekurak.pl/wprowadzenie-do-systemow-ids/>

IDS (ang. Intrusion Detection System) to urządzenie lub oprogramowanie, które monitoruje i analizuje ruch sieciowy w celu wyszukania podejrzanych aktywności wysyłając powiadomienia o ich znalezieniu. Ogólnym założeniem IDS jest wczesne poinformowanie zespołu bezpieczeństwa IT, że doszło lub może dojść do włamania w sieci. Informacje zawarte w takim alarmie będą na ogół zawierać:

- Źródło ataku - host
- Rodzaj ataku - skanowanie sieci, brute-force, eksfiltracja danych, itd.
- Typ komunikacji sieciowej - protokół, port, częstotliwość połączeń, itd.
- Cel ataku - urządzenie, host, konto, witryna, domena, itd.

Każdy system IDS powinien mieć jak najlepszą widoczność sieci. Im więcej ruchu jest w stanie obserwować, tym więcej zagrożeń jest w stanie wykryć. Wykrywanie włamań w narzędziach IDS jest technologią pasywną. To oznacza, że system rozpoznaje zagrożenie, ale nie ucina przepływu ruchu sieciowego. Tak jak wspominano, celem jest tylko identyfikacja i ostrzeżenie o widocznym zagrożeniu. Dlatego właśnie systemy IDS nie powinny być porównywane do technologii różnego rodzaju rozwiązań IPS (ang. Intrusion Prevention System) czy firewalli.

Podział systemów IDS

Istnieje kilka klasyfikacji IDS pod względem architektury oraz sposobu działania. Jednym z najbardziej podstawowych jest rozdzielenie systemów IDS na te, które są fizycznymi urządzeniami wpinanymi w sieć oraz na te, które działają jako software i są instalowane na serwerach i urządzeniach sieciowych jako agenty/sondy. Technologia pozostaje taka sama - analiza ruchu sieciowego i wykrywanie nieprawidłowości.

Innym kryterium podziału IDSów jest sposób detekcji. Rozróżniamy dwa główne:

1. **Oparty o sygnatury** – wykrywa zagrożenia i ataki na podstawie określonych wzorców, którymi mogą być np. przykład konkrete instrukcje i polecenia, sekwencja pakietów sieciowych, liczba 0 i 1 w pakiecie czy protokół poprzez niestandardowy port. Taka metoda może z wysoką skutecznością wykryć znane ataki i techniki często używane przez cyberprzestępco, ale trudno jest jej wykryć nowe zagrożenia. Charakteryzuje się także mniejszą ilością alarmów false-positive.
2. **Oparty o anomalie** – został wprowadzony w celu wykrywania nieznanych zagrożeń i nowego złośliwego oprogramowania. Ponieważ w ostatnich latach malware szybko ewoluje i zmienia swoje zachowanie, metoda ta jest dzisiaj bardziej skuteczna. W IDS opartym o wykrywanie anomalii sieciowych wykorzystuje się uczenie maszynowe do tworzenia zaufanych modeli aktywności i porównywanie wszystkiego z istniejącym modelem. Jeśli zachowanie jakiegoś urządzenia, aplikacji czy użytkownika odbiega od normy, to generowany jest alert. Metoda ta jest bardziej ogólna i wykrywa więcej zagrożeń, jednak cechuje się większą ilością zdarzeń false-positive.

Najlepsze systemy IDS na rynku wykorzystują obie metody detekcji i są nazywane hybrydowymi rozwiązaniami IDS.

Warto jeszcze przedstawić podział systemów IDS pod względem monitorowanej infrastruktury. Główne rodzaje to:

1. **NIDS** (ang. Network Intrusion Detection System) - takie systemy są najczęściej urządzeniem umiejscowionym w konkretnym punkcie w sieci w celu badania ruchu ze wszystkich urządzeń. Dobrym miejscem do takiego monitoringu jest główny firewall w sieci i analiza wszystkiego co przez niego przechodzi w obu kierunkach.
2. **HIDS** (ang. Host Intrusion Detection System) - działa na niezależnych hostach i urządzeniach sieciowych. Monitoruje wychodzące pakiety tylko z konkretnych urządzeń. Agent instalowany na maszynie wykonuje również migawkę plików systemowych przed i po analizowanej aktywności sieciowej i w razie potrzeby alarmuje o wykrytych nieprawidłowościach.
3. **PIDS** (ang. Protocol-based Intrusion Detection System) - monitoruje konkretny ruch sieciowy między serwerem a klientem. Zwykle używany jest do obserwowania zagrożeń na serwery www. Monitoruje wtedy strumień HTTP i HTTPS i sprawdza jego zawartość oraz weryfikuje czy ruch sieciowy nie zawiera innych zbędnych protokołów.
4. **APIDS** (ang. Application Protocol-based Intrusion Detection System) - identyfikuje włamania do konkretnych aplikacji, jak na przykład zdalne ataki na SQL.

Jakie zagrożenia wykrywają?

- **Skanowanie sieci** – każdy atak rozpoczyna się od [rozpoznania sieciowego](#), poznania topologii, aktywnych hostów, otwartych portów
- **Ataki DDoS** – widoczne w sieci jako ogromne zwiększenie ilości połączeń przychodzących do konkretnego hosta bądź usługi
- **Buffer Overflow** – widoczne w pakiecie sieciowym jako ogromna ilość losowych stringów lub liczb powodujące przepełnienie bufora przy próbie wpisania do pamięci
- **Ataki oparte o konkretne protokoły:**
 - [Ataki z wykorzystaniem PING](#)
 - [Ataki za pośrednictwem DNS](#)
 - [Ataki z wykorzystaniem LDAP](#)
 - [Exploitacja SMB](#)
- **Rozprzestrzenianie się malware** – najczęściej jest to ransomware próbujący rozprzestrzenić się na inne hosty w sieci i zaszyfrować jak największą ilość plików, przykładem jest znany atak WannaCry
- **Komunikacja z C2** – praktycznie każdy malware będzie chciał wcześniej czy później skomunikować się ze swoim serwerem Command & Control
- **Brute-Force** – wszystkie próby nieudanych uwierzytelnienia do zasobów sieciowych są widoczne dla systemów IDS i mogą być alarmowane
- **Infiltracja i eksfiltracja danych** – Pobieranie niebezpiecznych plików lub kradzież plików na zewnątrz z organizacji
- **Błędy konfiguracji sieci** – błędy takie jak na przykład asymetryczny routing czy otwarte porty mogą być wykorzystane przez hackerów
- **Kopanie kryptowalut, Torrenty, sieć TOR**

Można więc rzec, że systemy IDS są wszechstronnie uzdatnione. Są w stanie wykryć każde zagrożenie, które polega na specyficznej komunikacji sieciowej. Wszystkie malware działające lokalnie, zaszyte w pamięci komputera i nierożprzestrzeniające się na zewnątrz są dla IDS niezauważalne.

Istnieje wiele opinii specjalistów dotyczących przydatności IDS w organizacji. Coraz rzadziej przedsiębiorstwa kupują systemy IDS jako oddzielnego produktu. Bardziej skupiają się na tzw. „kombajnach” do bezpieczeństwa, które zawierają kilka funkcjonalności i jedną z nich jest właśnie wykrywanie zagrożenia w ruchu sieciowym. Chociaż system IDS nie zatrzymuje złośliwego oprogramowania, technologia ta nadal powinna mieć swoje miejsce w każdej większej sieci informatycznej. Z narzędzia IDS nie będzie jednak pożytku, jeśli do obsługi generowanych przez niego incydentów, nie przydzielimy wystarczająco dużo zasobów ludzkich (bądź ew. maszynowych). Wszystko to dlatego, gdyż sporą wadą systemów IDS jest generowanie dużej ilości alarmów false-positive. Dodatkowo osoby obsługujące incydenty, muszą posiadać niezbędną, minimalną wiedzę o środowisku, aby poprawnie zdiagnozować problemy. IDS nie powinien być porównywany z innymi technologiami do bezpieczeństwa operującymi na ruchu sieciowym. Szczególnie chodzi o firewall'e. Możemy wyobrazić sobie, że firewall działa jak brama chroniąca naszą posesję, a IDS to alarm przeciwwłamaniowy. I jeden i drugi system jest potrzebny, ale również i jeden i drugi system przestępca jest w stanie obejść 😊

IPS

<https://kapitanhack.pl/co-to-jest-hips/>

Z definicji HIPS to zainstalowane bezpośrednio na urządzeniu końcowym oprogramowanie, które monitoruje pojedynczy host pod kątem podejrzanej aktywności, analizując zdefiniowane zdarzenia występujące na tym hoście. HIPS ma za zadanie zapobiec włamaniom i działaniom malware poprzez monitorowanie zachowania procesów aplikacji i akcji jakie podejmują w środowisku. Pozwala to bez konieczności aktualizowania informacji o nowych zagrożeniach w bazie sygnatur, skutecznie wykrywać złośliwe oprogramowanie.

Systemy zapobiegania oparte na hostach są głównie używane do ochrony urządzeń końcowych. Po wykryciu szkodliwej aktywności narzędzie HIPS może podjąć zdefiniowany wcześniej szereg działań. Wysłać alarm, rejestrować aktywność malware w logu, resetować połączenia sieciowe, blokować ruch sieciowy z określonych adresów czy portów. Niektóre systemy HIPS mają możliwość wysyłania w czasie rzeczywistym dzienników szkodliwej aktywności i fragmentów kodu bezpośrednio do dostawy lub zewnętrznego SOC w celu analizy i ewentualnej identyfikacji malware.

Historycznie narzędzia klasy HIPS wywodzą się od zwykłego firewalla. Koncepcja jest podobna. Zapora sieciowa reguluje ruch wchodzący oraz wychodzący z komputera na podstawie zestawu reguł, a HIPS dopuszcza lub blokuje zmiany wykonywane na komputerze również na podstawie zdefiniowanych zasad.

Zasada działania – 3 podejścia

Pierwsze podejście w systemach klasy HIPS polega na sygnaturach i jest w rzeczywistości uzupełnieniem dwóch kolejnych mechanizmów detekcji. Oprogramowanie obronne używa tutaj znanych wzorców wirusów, robaków i trojanów. Skutecznie chroni przed znymi atakami, ale nie ochroni przed zero-day czy złośliwym kodem, który nie znajduje się w bazie sygnatur. Dlatego właściwie podstawą do podjęcia akcji jest **drugie** podejście. Polega ono na zdefiniowaniu tego, co nazwiemy linią bazową aktywności w systemie, a następnie porównywanie wszystkich zachowań i detekcja odchyleń od normy. HIPS szuka anomalii między innymi w przepustowości łączna, w użyciu CPU i RAM, w używanych portach i protokołach, w tworzeniu nowych procesów potomnych itp. Gdy aktywność wystąpiła poza dopuszczalnym zakresem – na przykład zdalna aplikacja próbuje łączyć się do hosta na zamkniętym porcie – następuje alarm lub blokada niektórych czynności. Anomalie, na które szczególną uwagę zwracają systemy HIPS to gdy nowy proces lub program:

- przejmuje kontrolę na inną aplikację bez zgody użytkownika i na przykład wysyła maila lub łączy się z witryną i pobiera złośliwy kod
- modyfikuje rejestr
- kończy działanie innych programów – na przykład antywirusa
- instaluje sterowniki czy pluginy
- ingeruje w pamięć innych procesów

Warto tutaj zaznaczyć, że narzędzia HIPS polegają tylko na zestawie reguł zdefiniowanych przez użytkownika lub dostarczonych przez producenta. Nie są wyposażone w algorytmy uczenia maszynowego. Przez są ciężkie do wdrożenia w organizacji i alarmują wiele zdarzeń „false-positive”.

Trzecia powszechna metoda wykrywania włamań w systemach obronnych HIPS wykorzystuje kontrolę protokołów w pakietach przychodzących do urządzenia. Również wymaga zdefiniowania, jak powinien wyglądać pakiet z protokołu DNS, SMTP czy HTTP i jakie wartości powinny być w nich zawarte. Analiza protokołów szuka odchyleń od stanu normalnego i może wskazać ewentualny atak. Metoda ta jest bardziej świadoma od poprzedniej i zwykle wykazuje mniej zdarzeń false-positive.

Network-based Intrusion Prevention System

Narzędzia cyberbezpieczeństwa pod akronimem NIPS działają według tej samej koncepcji co HIPS, z jednym tylko wyjątkiem. Monitorowanie i detekcja odbywają się tutaj od strony ruchu sieciowego, a nie od strony konkretnego hosta. Taki system przystosowany jest głównie do wykrywania ataków DDoS, ruchu C&C oraz podejrzanych zapytań HTTP czy DNS. Po instalacji w środowisku, NIPS tworzy w sieci zabezpieczone fizycznie strefy, które są czymś w rodzaju kwarantanny dla złośliwego ruchu.

Jest kilka rzeczy, o których warto pamiętać odnośnie systemu HIPS, a także NIPS.

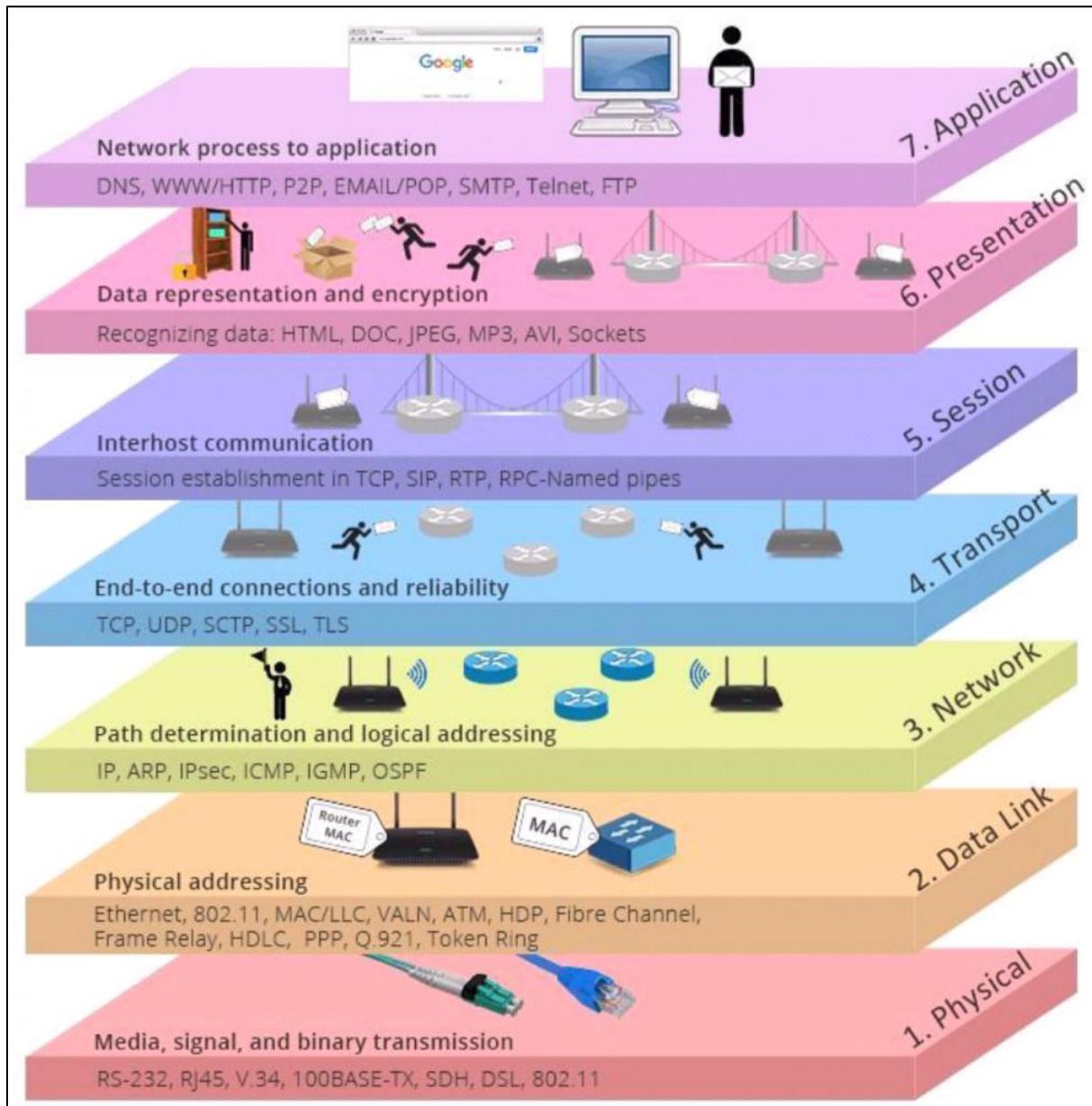
Przede wszystkim, jak przy większości takich artykułów przestrzegamy, że takie narzędzia nie są „złotym środkiem” i szczepionką na wszystko. Mogą być świetnym dodatkiem do innych warstw ochrony jak firewall, antywirus czy IDS, ale nie powinny próbować zastąpić istniejących technologii.

Po drugie, wdrożenie rozwiązania HIPS może być trudne i żmudne. Konfigurowanie właściwej linii bazowej często wymaga zrozumienia jak działają inne aplikacje w systemie i doświadczalne sprawdzanie wprowadzonych reguł. Z pewnością doświadczymy też dużej ilości fałszywych alarmów (false-positive) lub co gorsze, możemy przeoczyć zdarzenia, które powinny zostać wskazane jako anomalia (false-negative).

Często organizacje nie podejmują decyzji zakupu systemów HIPS, ponieważ wdrożenie go w rozległej infrastrukturze przekracza budżet na produkty bezpieczeństwa. Ciężko jest też przekonać Zarząd i dyrekcję, co do wartości dodanej jaką posiadają takie systemy w stosunku do popularnych i dużo tańszych antywirusów. Warto może pomyśleć, co firma może stracić, jeśli atak nastąpi i takiego systemu nie będzie. Jakie koszty wtedy zostaną poniesione i czy nie przekroczą one kosztów HIPS.

Firewall

Celem Firewalla jest upraszczając, skanowania każdego przychodzącego i wychodzącego pakietu danych, sprawdzania złośliwej lub nieodpowiedniej zawartości, a następnie zezwalanie lub odrzucanie na wprowadzenie pakietu danych do systemu firmy. Zapory sieciowe nowej generacji – NGFW mogą być oparte bądź na oprogramowaniu bądź sprzęcie i mogą posunąć się dalej niż zwykła ocena pakietów danych; dokonują głębokiej inspekcji pakietów, nie ograniczając się tylko do sprawdzenia numerów portów i typów protokołów. Umożliwiają kontrolę ruchu na poziomie warstwy aplikacji, zapobiegają włamaniom i mogą korzystać z różnych mechanizmów wykraczających poza funkcjonalność klasycznej zapory sieciowej. Pracę Next Generation Firewall i zabezpieczeń realizowanych na 7 warstwach sieci przedstawia poniższy schemat.



Warstwy sieci obsługiwane przez Next Generation Firewall

Jak historycznie rozwijały się Zapory Sieciowe?

Pierwsza generacja: packet filters Celem Firewalla jest upraszczając, skanowania każdego przychodzącego i wychodzącego pakietu danych, sprawdzania złośliwej lub nieodpowiedniej zawartości, a następnie zezwalanie lub odrzucanie na wprowadzenie pakietu danych do systemu firmy. Zapory sieciowe nowej generacji – NGFW mogą być oparte bądź na oprogramowaniu bądź sprzęcie i mogą posunąć się dalej niż zwykła ocena pakietów danych; dokonują głębskiej inspekcji pakietów, nie ograniczając się tylko do sprawdzenia numerów portów i typów protokołów. Umożliwiają kontrolę ruchu na poziomie warstwy aplikacji, zapobiegają włamaniom i mogą korzystać z różnych mechanizmów wykraczających poza funkcjonalność klasycznej zapory sieciowej. Pracę Next Generation Firewall i zabezpieczeń realizowanych na 7 warstwach sieci przedstawia poniższy schemat.

Druga generacja: „stateful” filters W latach 1989–1990 trzech kolegów z AT & T Bell Laboratories, Dave Presotto, Janardan Sharma i Kshitij Nigam opracowali drugą generację firewalli, nazywając je bramkami na poziomie obwodu (ang. circuit-level gateways). Zapory sieciowe drugiej generacji wykonują pracę poprzedników, ale działają do warstwy 4 (warstwa transportowa) modelu OSI. Osiąga się to poprzez zatrzymanie pakietów, dopóki nie będzie wystarczającej ilości informacji, aby ocenić ich stan. Firewall rejestruje wszystkie połączenia przechodzące przez niego i określa, czy pakiet jest początkiem nowego połączenia, częścią istniejącego połączenia, czy nie jest częścią żadnego połączenia. Chociaż reguły statyczne są nadal używane, mogą teraz zawierać stan połączenia jako jedno z kryteriów testowych.

Trzecia generacja: application layer Marcus Ranum, Wei Xu i Peter Churchyard opracowali trzecią generację firewalla znaną jako Firewall Toolkit (FWTK). W czerwcu 1994 roku Wei Xu rozszerzył FWTK o ulepszenie jądra filtra IP i niewidoczne gniazda. Była ona znana jako pierwsza niewidoczna zapora aplikacji, udostępniona jako produkt komercyjny w zaufanych systemach informatycznych (Trusted Information Systems). Zapora sieciowa Gauntlet została oceniona jako jedna z najlepszych zapór sieciowych w latach 1995–1998. Kluczową zaletą filtrowania warstw aplikacji jest to, że może ona „zrozumieć” określone aplikacje i protokoły (takie jak protokół FTP, system nazw domen (DNS) lub protokół HTTP (Hypertext Transfer Protocol)). Jest to użyteczne, ponieważ jest w stanie wykryć, czy niechciana aplikacja lub usługa próbuje ominąć zaporę sieciową za pomocą protokołu na dozwolonym porcie lub wykryć, czy protokół jest nadużywany w jakikolwiek szkodliwy sposób. Począwszy od 2012 roku, tak zwana zapora następnej generacji (NGFW) to nic innego jak „szersza” lub „głębsza” inspekcja na stosie aplikacji. Na przykład istniejąca funkcja głębskiej inspekcji pakietów w nowoczesnych zaporach sieciowych może zostać rozszerzona o takie funkcje jak:

- Systemy zapobiegania włamaniom (IPS)
- Integracja zarządzania tożsamościami użytkowników (poprzez powiązanie identyfikatorów użytkowników z adresami IP lub MAC dla „reputacji”)
- Zapora aplikacji WWW (WAF).

Co to jest urządzenie Unified Threat Management (UTM)?

Chociaż NGFW może być programowy lub sprzętowy, urządzenia UTM są zawsze sprzętowe. Ma to zarówno wady jak i zalety. Niewielką wadą urządzenia UTM jest to, że aby funkcjonować, musi być podłączone do sieci głównej. Zaletami urządzenia UTM jest szeroka gama funkcji, które może wykonywać z tej pozycji:

- Równoważenie obciążenia dla sieci
- Przeciwdziałanie wyciekom danych
- Zapobieganie utracie danych
- Zapobieganie wirusom
- Realizowanie funkcji bramy antyspamowej
- Wykrywanie i zapobieganie włamaniom do sieci
- Raportowanie urządzeń
- Bezpieczeństwo poczty e-mail
- Filtrowanie adresów URL
- Filtrowanie zawartości pakietów
- Ochrona sieci bezprzewodowej
- Zakończenie wirtualnej sieci prywatnej
- Realizowanie funkcji zapory aplikacji internetowych
- Ograniczanie DDOS

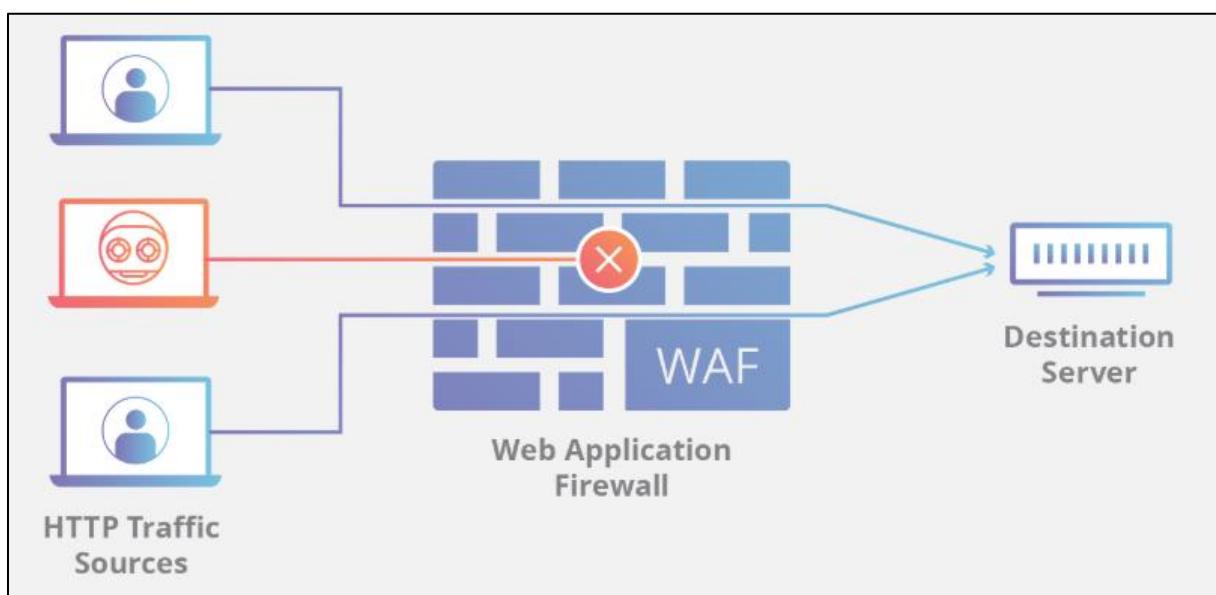
Ta lista narzędzi oznacza, że zasadniczo urządzenie UTM obejmuje wszystkie funkcje NGFW. Możliwe jest uzyskanie wszystkich tych usług od niezliczonych dostawców i aplikacji. Zaletą urządzenia UTM jest jednak to, że wszystkie dane są skoncentrowane i mogą być przeglądane całkowicie, zapewniając lepszy przegląd stanu wykrywania zagrożeń sieciowych w czasie rzeczywistym.

WAF

WAF, czyli Web Application Firewall to system bezpieczeństwa przystosowany do ochrony aplikacji internetowych poprzez monitorowanie i filtrowanie ruchu HTTP między aplikacją a Internetem. WAF operuje na warstwie 7 (w modelu OSI) i nie jest niestety przeznaczony do obrony przed wszystkimi rodzajami ataków na aplikacje internetowe. Zwykle skutecznie chroni przed takimi atakami jak SQL Injection, Cross Site Scripting, Directory Traversal czy Command Injection.

Wdrażając WAF umieszczamy pewnego rodzaju tarczę między aplikacją a Internetem. Podczas gdy, serwer proxy chroni tożsamość komputera klienckiego działając jako pośrednik, WAF działa jako reverse-proxy i chroni tożsamość serwera przed poznaniem przez klientów.

Działanie takiego systemu z pewnością można porównać do klasycznego [firewalla](#). Opiera się ono o zestaw konfigurowalnych reguł nazywanych politykami, które mają za zadanie chronić luki w zabezpieczeniach aplikacji i odfiltrowywać szkodliwy ruch. Mocną stroną systemów WAF jest łatwość i szybkość implementacji nowych reguł oraz pewnego rodzaju automatyzacja i samodostosowywanie. Na przykład podczas ataku DDoS możliwa jest szybka zmiana ilości dopuszczalnych połączeń na sekundę.



Zasada działania

Web Application Firewall może być wdrażany jako osobne urządzenie sprzętowe lub wtyczka instalowana na serwerze webowym. Przechwytuje wszystkie żądania HTTP i analizuje każde z nich, zanim dotrą do serwera WWW w celu przetworzenia. Skonfigurowane reguły analizują żądania GET i POST i szukają w nich złośliwego ruchu.

W zależności od opcji konfiguracji narzędzia możliwe jest blokowanie ruchu do aplikacji, zmuszenie odwiedzającego do weryfikacji i wypełnienia testu CAPTCHA czy nawet symulacja ataku na serwerze. Dzięki temu, atakujący nie zostanie wpuszczony do serwera hostującego aplikację.

Działanie systemu WAF opiera się zazwyczaj na jednym z trzech modeli:

- „**Blacklist**” lub „**Negative Security Model**” – wykorzystanie sygnatur do ochrony witryny przed znymi konkretnymi atakami, które mogą wykorzystywać podatności w aplikacji internetowej
- „**Whitelist**” lub „**Positive Security Model**” – wykorzystanie sygnatur oraz logiki do zezwolenia na ruchu tylko jeśli spełnia określone kryteria, na przykład zezwolenie tylko na żądania HTTP GET z określonego adresu URL i blokowanie wszystkiego innego
- „**Hybrid Security Model**” – połączenie możliwości dwóch poprzednich

Niektóre konfigurowalne opcje pozwalają na wykonywanie akcji takich jak blokowanie sesji, blokowanie adresu IP, blokowanie lub wylogowywanie użytkownika.

Różne możliwości wdrożenia

WAF można wdrożyć w infrastrukturze na co najmniej 3 różne sposoby, przy czym każdy z nich ma swoje zalety i wady:

- **Network-based WAF** – oparte na własnym, niezależnym sprzęcie, WAF jest instalowany lokalnie co minimalizuje opóźnienia, ale też wymaga zasobów do przechowywania danych oraz konserwacji, jest opcją najdroższą
- **Host-based WAF** – zintegrowanie z serwerem hostującym aplikacje, zużywane są zasoby lokalne serwera, opcja ta oferuje najwięcej możliwości dostosowywania WAF do własnych potrzeb, jest jednak trudna w implementacji i konserwacji, wymaga poświęcenia czasu inżynierów w organizacji, tańsza niż opcja numer 1
- **Cloud-based WAF** – wdrożenie rozwiązania w chmurze, jest to opcja najprostsza oraz najtańsza, instalacja polega zazwyczaj na zmianie wpisu w DNS i przekierowania ruchu, takie rozwiązanie jest konsekwentnie aktualizowane i konserwowane przez osoby trzecie, wadą jest to, że pełna odpowiedzialność za system przekazana jest stronie trzeciej i niektóre elementy WAF pozostają „czarną skrzynką” dla organizacji

Rozwiązania WAF skutecznie zapobiegają atakom skierowanym na aplikacje internetowe. Trzeba pamiętać, że rozwiązania te wciąż ewoluują oraz to, że żadne narzędzie nie jest w stanie wyeliminować wszystkich słabych stron aplikacji. Często konieczne jest użycie więcej niż jednego rodzaju zabezpieczeń.

Typowym podejściem jest połączenie WAF z narzędziem DAST (Dynamic Application Security Testing). Narzędzia te są przeznaczone do sprawdzania luk w zabezpieczeniach w uruchomionych w Internecie aplikacjach. Robią to poprzez wysyłanie żądań imitujących aktywność hackera.

Web Application Firewall oprócz ochrony zapewnia dodatkowe funkcje takie jak buforowanie, kompresja, równoważenie obciążenia, akceleracja SSL czy zestawianie połączeń, które dodatkowo zwiększą wydajność i niezawodność witryny.

SOAR

<https://www.microsoft.com/pl-pl/security/business/security-101/what-is-soar>

<https://www.nomios.pl/materiały/czym-jest-soar/>

SNORT

Wprowadzenie

Snort jest otwartoźródłowym systemem wykrywania intruzów wdzierających się z sieci (ang. network intrusion detection system, IDS), może też służyć jako system zabezpieczania przed intruzami wdzierającymi się z sieci (ang. network intrusion prevention system, IPS). Oprogramowanie to jest rozwijane przez Cisco. Jednak ważnym elementem zaufania do tego systemu jest jego otwartoźródłowość - każdy ma możliwość sprawdzenia, że zastosowanie Snorta nie spowoduje wprowadzenia intruza bocznymi drzwiami przez zastosowanie tego narzędzia. Dodatkowo duża popularność narzędzia sprawia, że w tym przypadku otwartoźródłowość sprzyja zmniejszaniu zagrożenia.

Snort może działać w wielu rodzajach strumieni danych:

- na ruchu kierowanym do maszyny,
- na ruchu, który dociera do maszyny,
- na ruchu tranzytowym pomiędzy interfejsami i
- na ruchu tranzytowym poprzez segment sieci.

Na takim ruchu może on też wykonywać różnego rodzaju funkcje:

- może działać jako system wykrywania intruzów (IDS), gdy tylko wysyła alarmy,
- może działać jako system zabezpieczania przed intruzami (IPS), gdy oprócz wysyłania alarmów, również nie dopuszcza do przekazywania ruchu o określonej, niebezpiecznej charakterystyce.

Instalacja i początkowa konfiguracja

Na dobry początek warto zainstalować Snorta na maszynie wirtualnej:

```
# apt-get install snort snort-doc
```

Podczas instalacji otrzymamy zapytanie o sieć, na której będziemy wykonywać badania. W związku z tym, że będziemy tego obrazu używali w różnych sieciach, pole to należy wyczyścić i odpowiednio ustawić zmienną HOME_NET w pliku /etc/snort/snort.conf.

W tym celu po zakończeniu instalacji otwieramy wspomniany plik i zauważamy, że edytowanie wspomnianej zmiennej nie jest właściwe i należy otworzyć plik /etc/snort/snort.debian.conf, gdzie należy zmiennej DEBIAN_SNORT_HOME_NET nadać wartość opisującą adres sieci, przed atakami z której chcemy się bronić. Możemy ją wyczytać np. za pomocą polecenia ifconfig. Widząc tam zapis

```
inet 10.2.7.46 netmask 255.255.240.0 broadcast 10.2.15.255
```

wpisujemy do zmiennej wartość 10.2.0.0/20, bo wskazuje on, że właśnie 20 bitów zajmuje część adresu określająca adres sieci.

Pierwsze uruchomienie

Możemy zacząć od tego, żeby sprawdzić, czy plik konfiguracyjny Snorta jest prawidłowy (na pewno jest prawidłowy po zainstalowaniu, ale warto zobaczyć, jak w takiej sytuacji wygląda działanie programu):

```
# snort -T -i enp0s3 -c /etc/snort/snort.conf 2>&1 | less
```

Flaga -T oznacza właśnie testowe wykonanie, -i pozwala wskazać, jakiego interfejsu oglądanie nas interesuje, zaś opcja -c pozwala wskazać plik konfiguracyjny. Po uruchomieniu tego polecenia ukaże nam się bardzo długa ściana tekstu, która zawiera raport z poszczególnych etapów wczytywania reguł zakończony podsumowaniem (też długim) zaczynającym się od opisu:

```
4057 Snort rules read
3383 detection rules
0 decoder rules
0 preprocessor rules
3383 Option Chains linked into 932 Chain Headers
```

który wskazuje, ile reguł zostało wczytane oraz ile z nich jest związane z wykrywaniem nieprawidłowości, ile z odkodowywaniem danych, a ile ze wstępny przetwarzaniem. Wypisane informacje kończą się opisem wersji Snorta oraz oprogramowania, z którego on korzysta.

Alerty Snorta

Cała siła programu Snort leży w regułach, które można pogrupować w zestawy. Po krótkim przyjrzeniu się plikowi /etc/snort/snort.conf łatwo stwierdzimy, że do pliku konfiguracyjnego dotaczane są liczne pliki z katalogu /etc/snort/rules/, zawierającego właśnie różne zestawy reguł pogrupowane „tematycznie”. Można sobie rzucić okiem, jakie zestawy są dostępne, i jak wyglądają poszczególne wpisy.

Zwykle diagnozowanie działania sieci, również z użyciem Snorta, zaczynamy od ustawienia interfejsu sieciowego karty w tryb promiscuous. W trybie tym karta przekazuje do oprogramowania wszystkie pakiety, które do niej docierają - również te, które nie są do niej adresowane. Co prawda w dzisiejszych czasach ruch pakietów adresowanych do innych interfejsów niż interfejs na końcu kabla jest drastycznie ograniczony przez przełączniki, ale nie zawsze to działa (NB. warto zgłosić administratorom, gdy na interfejsie pojawiają się pakiety do niego nie adresowane). Przełączenie karty w tryb promiscuous uzyskujemy za pomocą polecenia:

```
# ip link set enp0s3 promisc on
lub
# ifconfig enp0s3 -promisc
```

Możemy teraz uruchomić Snorta w celu określenia, czy nie dochodzi do naszej maszyny jakiś podejrzany ruch:

```
# snort -d -l /var/log/snort/ -h 10.2.0.0/20 -A console -c /etc/snort/snort.conf
```

Tutaj opcja -d wskazuje, że Snort jest używany w trybie systemu wykrywania intruzów (ang. Network Intrusion Detection System, NIDS). W trybie tym widzimy tylko informacje pochodzące z ruchu o podejrzanych charakterystykach (np. pakiety o niepoprawnej strukturze). Opcja -l wskazuje, gdzie są umieszczane pliki z zebranymi przez Snorta informacjami. Dodanie -h 10.2.0.0/20 powoduje, że widzimy nieco mniej informacji na ekranie, a podejrzany ruch z odległych maszyn umieszczany jest w katalogach o nazwach zgodnych z nazwami tych maszyn, zaś -A console powoduje, że dostajemy raporty na ekranie terminala. Możemy sobie chwilę poobserwować ruch, żeby się nieco oswoić z rodzajem uzyskiwanej informacji. Warto raz na jakiś czas naciągnąć na dłużej enter, żeby sobie oczyścić ekran z nadmiaru informacji.

Najbardziej typowym sposobem badania komputerów w sieci jest wyłanie do maszyny pinga. Wykonajmy z jakiejś innej maszyny niż ta, na której działa Snort, polecenie:

```
# ping 10.2.7.46
```

Zapewne nic tutaj nie zobaczymy, bo reguły Snorta są tak napisane, aby nie robić alarmu w przypadku ruchu zasadniczo nieszkodliwego. Natomiast jeśli spróbujemy wysłać niestandardowe pakiety ping, to ujrzymy nietrywialny efekt działania Snorta. Po wysłaniu pakietu ping (technicznie jest to pakiet ICMP echo request) o rozmiarze 4096

```
# ping -s 4096 10.2.7.46
```

dostaniemy następujące trzy sygnały od Snorta:

```
12/04-20:00:39.553493 [**] [1:480:5] ICMP PING speedera [**] [Classification: Misc activity] [Priority: 3] {ICMP} 10.2.1.95 -> 10.2.7.46  
12/04-20:00:39.553493 [**] [1:499:4] ICMP Large ICMP Packet [**] [Classification: Potentially Bad Traffic] [Priority: 2] {ICMP} 10.2.1.95 -> 10.2.7.46  
12/04-20:00:39.553669 [**] [1:499:4] ICMP Large ICMP Packet [**] [Classification: Potentially Bad Traffic] [Priority: 2] {ICMP} 10.2.7.46 -> 10.2.1.95
```

Jak to zwykle bywa, wpisy zaczynają się od znaczników czasowych. Ciąg [**] to specyficzny separator używany przez Snorta. Następny znacznik, na przykład [1:480:5], wskazuje, jakie moduły wewnętrzne Snorta zajmowały się przetwarzaniem prowadzącym do danego alarmu (pierwsza 1, zob. plik etc/generators w źródłach Snorta), jaka reguła została użyta (patrzmy na wartość sid w regule) oraz w jakiej wersji (patrzmy na wartość rev w regule).

Pochodzenie alarmu możemy w ogromnej większości przypadków (nie dotyczy to reguł preprocesora) stwierdzić, wykonując polecenie w stylu:

```
# grep sid:499 /etc/snort/rules/*
```

Powyższa instrukcja szybko nam też wyjaśni pochodzenie następnego napisu - komentarza do alertu. Następnie po separatorze [**] widać określenie, do jakiej kategorii zdarzeń został dany alert przydzielony. Z każdą kategorią zdarzeń związany jest domyślny priorytet zwykle konfigurowany w pliku /etc/snort/classification.config. Można jednak te wartości nadpisywać w regułach.

Końcowy zapis {ICMP} 10.2.7.46 -> 10.2.1.95 jest zapisem przepisany na początku reguły (wskażanie protokołu niższej warstwy oraz adresów pochodzenia i docelowego pakietu; czasami rozszerzone o numer portu).

Inny przykład raportu, jaki daje Snort, uzyskamy, wydając (na innej maszynie wirtualnej) polecenie:

```
# nmap -A 10.2.7.46
```

Wtedy uzyskamy raport postaci:

```
12/04-19:50:41.178872 [**] [1:1418:11] SNMP request tcp [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 10.2.1.95:35957 -> 10.2.1.95:161  
12/04-19:50:41.197157 [**] [1:1421:11] SNMP AgentX/tcp request [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 10.2.1.95:35957 -> 10.2.1.95:705  
12/04-19:50:42.150598 [**] [1:1228:7] SCAN nmap XMAS [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 10.2.1.95:47570 -> 10.2.1.95:1  
12/04-19:50:42.150779 [**] [1:485:4] ICMP Destination Unreachable Communication Administratively Prohibited [**] [Classification: Misc activity] [Priority: 3] {ICMP} 10.2.1.95 -> 10.2.1.95
```

Wskazujący, że program nmap do określenia systemu operacyjnego na komputerze, który jest badany, używa protokołu SNMP, następnie wykonuje skanowanie portów, aby zakończyć proces wysłaniem niestandardowego pakietu ICMP.

Reguły Snorta

Spróbujmy teraz napisać własną regułę Snorta. Umieścimy ją w pliku /etc/snort/rules/local.rules i nadamy jej taką postać:

```
alert tcp $EXTERNAL_NET any ->$HOME_NET 23 (msg: "Attempt to telnet from external net"; sid:10000002; rev:1;)
```

Pamiętajmy, że reguła musi być zapisana w jednym wierszu.

Teraz próba wykonania polecenia telnet z adresem naszej maszyny wirtualnej spowoduje wyświetlenie się odpowiedniego alarmu Snorta. Jednak taki alarm może być dla nas niezadowalający: brakuje klasyfikacji i priorytet jest bardzo wysoki. Dlatego poprawmy regułę tak:

```
alert tcp $EXTERNAL_NET any ->$HOME_NET 23 (msg: "Attempt to telnet from external net"; sid:10000002; rev:2; classtype:attempted-user)
```

Tym razem próba telnetu da nam już wskazanie, że alarm należy do określonej klasy, a także wskaże sensowny jej priorytet. Warto zwrócić uwagę, że przy okazji zmiany zwiększyła się też wartość pola rev. Zwiększyliśmy je o jeden, żeby wskazać, iż zaszła zmiana i w obiegu mogą być alerty pochodzące z różnych wersji tej reguły, a zatem prawdopodobnie mające różne znaczenie.

Reguły Snorta pozwalają na badanie wielu charakterystyk pakietów, jakie trafiają do komputera. Proszę przyjrzeć się regułom z plików /etc/snort/rules/web-*, aby zorientować się, jakie rodzaje alarmów są tworzone dla aplikacji webowych. Warto też obejrzeć dokumentację ze strony <https://www.snort.org>.

Snort bywa umieszczany jako część dedykowanego systemu operacyjnego służącego do wykrywania intruzów. Przykładem takiego rozwiązania jest system [Security Onion](#).

Gdy Snort jest używany jako sonda IDS/IPS, która bada ruch w segmencie sieciowym, jest uruchamiany wraz z odpowiednimi rozwiązaniami sprzętowymi takimi jak [urządzenia TAP](#) lub na końcu połączenia przez [port lustrzany](#).

Strona Snorta oferuje kilka rodzajów reguł (zob. <https://www.snort.org/downloads/#rule-downloads>):

- reguły tworzone przez społeczność, do których dostęp jest bezpłatny i nieograniczony;
- reguły tworzone przez ekspertów bezpieczeństwa z zespołu Talos; reguły te są bezpłatne, ale wymagają zarejestrowania na stronie Snorta;
- reguły komercyjne, za które trzeba płacić, ale zapewniają szybsze reagowanie na potencjalne luki w bezpieczeństwie.

Są też dostępne inne subskrypcje komercyjne, oferujące takie na przykład firma Cisco.

Suricata

<https://www.it-and-security.pl/blog/2025/09/03/039-suricata-opnsense-pub/>

Suricata to potężny, wielowątkowy silnik wykrywania i zapobiegania włamaniom (IDS/IPS), a także system monitorowania bezpieczeństwa sieci (NSM). Opracowana przez Open Information Security Foundation (OISF), jest oprogramowaniem "open source", które w ciągu ostatnich lat zyskało ogromną popularność dzięki swojej wydajności i elastyczności.

Jako IDS, Suricata działa w trybie pasywnym, skanując pakiety danych i generując alerty, gdy wykryje wzorce pasujące do znanych zagrożeń, takich jak złośliwe oprogramowanie, skanowanie portów czy próby wykorzystania luk w zabezpieczeniach. Pozwala to administratorom na monitorowanie potencjalnie szkodliwego ruchu bez jego blokowania.

Natomiast w trybie IPS, Suricata staje się aktywnym elementem zabezpieczeń, który nie tylko wykrywa, ale także aktywnie blokuje niebezpieczny ruch w czasie rzeczywistym. Dzięki zastosowaniu rozbudowanych zestawów reguł, może na bieżąco odrzucać pakiety, które stanowią zagrożenie dla bezpieczeństwa sieci, skutecznie powstrzymując ataki, zanim dotrą do docelowych systemów.

Jedną z kluczowych zalet Suricaty jest jej zdolność do analizy ruchu na różnych warstwach protokołów (od warstwy 2 do 7), a także obsługa protokołów na poziomie aplikacji, co pozwala na identyfikację bardziej złożonych i ukrytych zagrożeń.

3. Reguły

Sercem każdego systemu IDS/IPS, a w szczególności Suricaty, są reguły (ang. *rules*). To one definiują, co silnik systemu ma uznać za potencjalne zagrożenie. Reguły to zestawy instrukcji, które określają wzorce do wykrycia w ruchu sieciowym. Mogą one dotyczyć prostych sygnatur (np. charakterystyczne ciągi znaków w pakietach danych), a także bardziej złożonych warunków (np. podejrzane zachowanie protokołu, nietypowy rozmiar pakietu, czy próby dostępu do złośliwych domen).

Suricata wykorzystuje reguły w formatach kompatybilnych z innymi systemami, co pozwala na korzystanie z szerokiej gamy dostępnych, często aktualizowanych zestawów reguł. Do najważniejszych należą:

Emerging Threats Open (ET Open): Powszechnie używany, darmowy zbiór reguł, opracowywany przez społeczność, która na bieżąco reaguje na nowe zagrożenia.

Proofpoint Emerging Threats Pro (ET Pro): Płatna, profesjonalna wersja zestawu, oferująca bardziej rozbudowane i częściej aktualizowane reguły, które wykrywają najnowsze zagrożenia, w tym te typu zero-day. **Zalecamy używanie tych reguł.**

OOT (Official OISF Trust Rules): Reguły rozwijane przez twórców **Suricaty**, które koncentrują się na najbardziej krytycznych i powszechnych zagrożeniach.

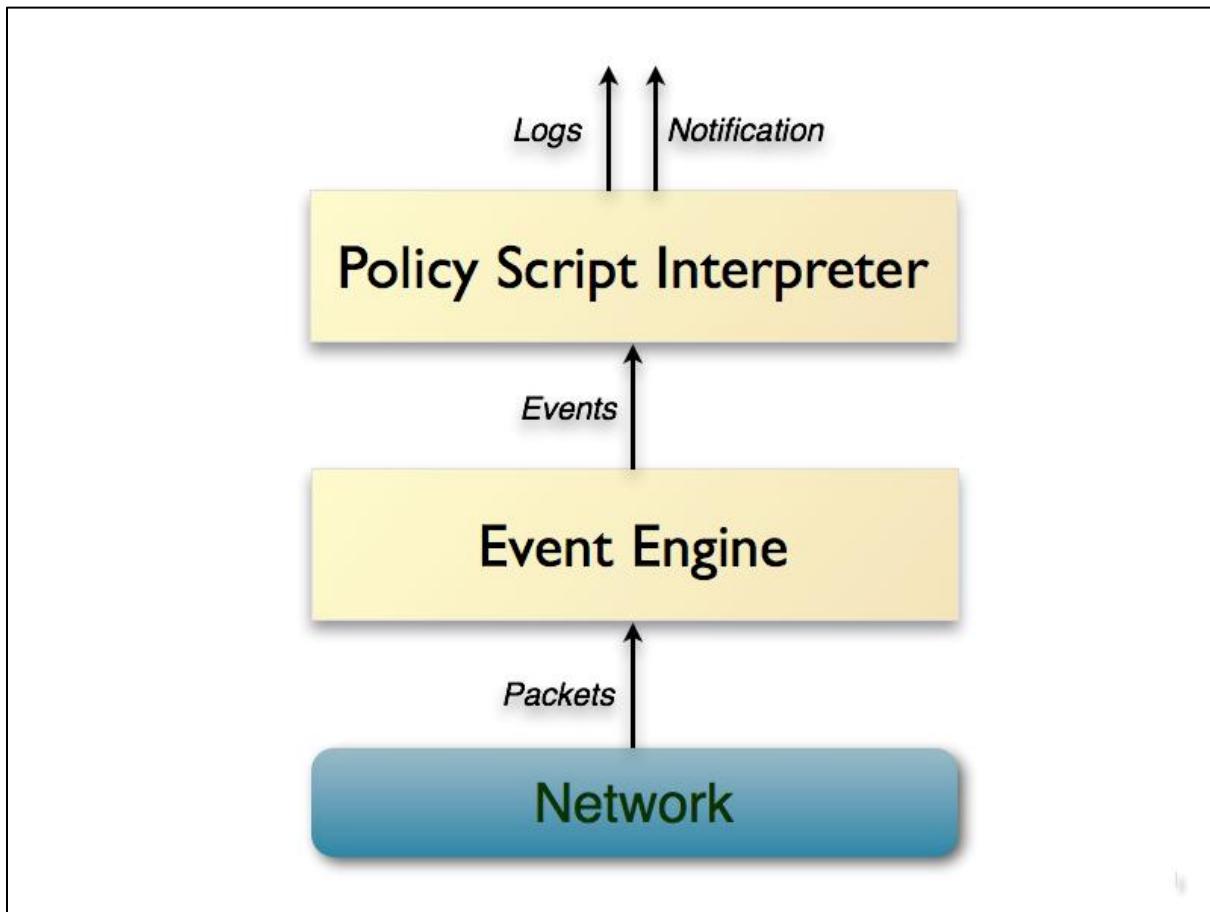
Zeek

<https://docs.zeek.org/en/current/about.html>

Zeek jest pasywnym, otwartoźródłowym analizatorem ruchu sieciowego. Wielu operatorów używa Zeeka jako monitora bezpieczeństwa sieci, czyli NSM, aby wspierać dochodzenia dotyczące podejrzanej lub złośliwej aktywności. Zeek obsługuje także szeroki zakres zadań analizy ruchu wykraczających poza obszar bezpieczeństwa, w tym pomiary wydajności i rozwiązywanie problemów. Pierwszą korzyścią, jaką nowy użytkownik uzyskuje z Zeekiem, jest rozbudowany zestaw logów opisujących aktywność sieciową. Logi te obejmują nie tylko kompleksowy zapis każdego połączenia zaobserwowanego na łączu, ale również transkrypcje warstwy aplikacji. Obejmują one wszystkie sesje HTTP wraz z żądanymi URI, kluczowymi nagłówkami, typami MIME i odpowiedziami serwera, zapytania DNS wraz z odpowiedziami, certyfikaty SSL, kluczową treść sesji SMTP i wiele więcej. Domyslnie Zeek zapisuje wszystkie te informacje w dobrze ustrukturyzowanych plikach logów rozdzielanych tabulatorami lub w formacie JSON, nadających się do dalszego przetwarzania przy użyciu zewnętrznego oprogramowania. Użytkownicy mogą też wybrać, aby zewnętrzne bazy danych lub produkty SIEM pobierały, przechowywały, przetwarzaly i prezentowały dane na potrzeby ich odpytywania. Oprócz logów Zeek ma wbudowaną funkcjonalność do szeregu zadań analizy i detekcji, w tym wyodrębniania plików z sesji HTTP, wykrywania złośliwego oprogramowania poprzez integrację z zewnętrznymi rejestrarami, raportowania podatnych wersji oprogramowania zaobserwowanych w sieci, identyfikowania popularnych aplikacji webowych, wykrywania ataków brute force na SSH, weryfikowania łańcuchów certyfikatów SSL i wielu innych. Poza dostarczaniem tak potężnej funkcjonalności od razu po uruchomieniu Zeek jest w pełni konfigurowalna i rozszerzalna platformą do analizy ruchu. Zeek udostępnia użytkownikom dziedzinowy, zupełny w sensie Turinga język skryptowy do opisywania dowolnych zadań analitycznych. Język Zeek można traktować jako dziedzinowy odpowiednik Pythona lub Perla: podobnie jak w Pythonie system zawiera duży zestaw gotowych funkcji, czyli biblioteką standardową, a jednocześnie użytkownicy mogą wykorzystywać Zeeka w nowych zastosowaniach, pisząc własny kod. W istocie wszystkie domyślne analizy Zeeka, w tym logowanie, są realizowane przez skrypty; żadne konkretne analizy nie są na stałe zaszyte w jądrze systemu. Zeek działa na powszechnie dostępnych podzespołach, a więc stanowi niskokosztową alternatywę dla drogich, właściwościowych rozwiązań. Pod wieloma względami Zeek przewyższa możliwości innych narzędzi do monitorowania sieci, które zwykle pozostają ograniczone do niewielkiego zestawu na stałe zakodowanych zadań analitycznych. Zeek nie jest klasycznym systemem wykrywania włamań opartym na sygnaturach, czyli IDS; choć wspiera również taką standardową funkcjonalność, język skryptowy Zeeka ułatwia znacznie szersze spektrum bardzo różnych podejść do wykrywania złośliwej aktywności. Obejmują one detekcję nadużyć semantycznych, detekcję anomalii oraz analizę behawioralną. Bardzo wiele różnych instytucji wdraża Zeeka do ochrony swojej infrastruktury, w tym wiele uniwersytetów, laboratoriów badawczych, centrów superkomputerowych, społeczności otwartej nauki, dużych korporacji oraz agencji rządowych. Zeek jest ukierunkowany na szybkie monitorowanie sieci o dużym wolumenie ruchu, a rosnąca liczba miejsc używa go obecnie do monitorowania sieci 10GE, przy czym część z nich przechodzi już na łączę 100GE. Zeek radzi sobie w środowiskach o wysokich wymaganiach wydajnościowych dzięki obsłudze skalowalnego równoważenia obciążenia. Duże ośrodki zwykle uruchamiają klastry Zeeka, w których szybki element front-end pełniący rolę load balansera rozdziela ruch na odpowiednią liczbę komputerów back-end, z których każdy uruchamia dedykowaną instancję Zeeka na przypisanym mu wycinku ruchu. Centralny system menedżera koordynuje proces, synchronizując stan pomiędzy węzłami back-end i zapewniając operatorom centralny interfejs zarządzania do konfiguracji oraz dostępu do zagregowanych logów. Zintegrowane środowisko zarządzania Zeekiem, ZeekControl, obsługuje takie konfiguracje klastrowe od razu po uruchomieniu. Funkcje klastrowe Zeeka wspierają zarówno konfiguracje na jednym systemie, jak i na wielu systemach. To część przewag skalowalności Zeeka. Na przykład administratorzy mogą skalować Zeeka w ramach jednego systemu tak długo, jak to możliwe, a następnie w razie potrzeby w przejrzysty sposób dodać kolejne systemy. W skrócie Zeek jest zoptymalizowany pod kątem interpretowania ruchu sieciowego i generowania logów na podstawie tego ruchu. Nie jest zoptymalizowany pod kątem dopasowywania bajtów, a użytkownicy poszukujący podejść opartych na detekcji sygnaturowej lepiej skorzystają z systemów wykrywania włamań takich jak Suricata. Zeek nie jest też analizatorem protokołów w rozumieniu Wiresharka, który stara się pokazać każdy element ruchu na poziomie ramek, ani systemem do przechowywania ruchu w postaci przechwyconych pakietów PCAP. Zamiast tego Zeek znajduje się w korzystnym środku, dostarczając zwarte, a jednocześnie bardzo wierne logi sieciowe, co pozwala lepiej rozumieć ruch i sposób wykorzystania sieci. Dlaczego Zeek? Zeek oferuje wiele zalet dla zespołów bezpieczeństwa i sieci, które chcą lepiej rozumieć, w jaki sposób używana jest ich infrastruktura. Zespoły bezpieczeństwa zazwyczaj opierają się na czterech rodzajach źródeł danych, gdy próbują wykrywać i reagować na podejrzana oraz złośliwą aktywność. Obejmują one źródła zewnętrzne, takie jak organy ścigania, partnerzy oraz komercyjne lub non-profit organizacje wywiadu o zagrożeniach, dane sieciowe, dane infrastruktury i aplikacji, w tym logi ze środowisk chmurowych, oraz dane z punktów końcowych. Zeek jest przede wszystkim platformą do zbierania i analizowania drugiej formy danych, czyli danych sieciowych. Wszystkie cztery są jednak ważnymi elementami programu każdego zespołu bezpieczeństwa. Przyglądając się danym pochodzący z sieci, analitycy mają do dyspozycji cztery typy

danych. Zgodnie z paradygmatem network security monitoring są to pełna treść, dane transakcyjne, treść wyodrębniona oraz dane alertowe. Korzystając z tych typów danych można odpowiednio rejestrować ruch, streszczać ruch, wyodrębniać z ruchu treść, lub być może dokładniej wyodrębniać treść w postaci plików, oraz oceniać ruch. Kluczowe jest zbieranie i analizowanie czterech typów danych monitorowania bezpieczeństwa sieci. Pytanie sprowadza się do ustalenia najlepszego sposobu osiągnięcia tego celu. Na szczęście Zeek jako platforma NSM umożliwia zbieranie co najmniej dwóch, a pod pewnymi względami trzech z tych form danych, mianowicie danych transakcyjnych, treści wyodrębnionej oraz danych alertowych. Zeek jest najbardziej znany z danych transakcyjnych. Domyślnie, gdy zostanie uruchomiony i otrzyma polecenie obserwowania interfejsu sieciowego, Zeek wygeneruje kolekcję związkowych, bardzo wiernych i bogato adnotowanych logów transakcyjnych. Logi te opisują protokoły oraz aktywność zaobserwowaną na łączu w sposób pozbawiony ocen i neutralny względem polityk. Ta dokumentacja poświęci znaczną ilość czasu na opis najczęściej spotykanych plików logów Zeeka, aby czytelnicy oswiili się z formatem i nauczyli się stosować je w swoim środowisku. Zeek potrafi też łatwo wycinać pliki z ruchu sieciowego dzięki możliwościom ekstrakcji plików. Analitycy mogą następnie wysłać te pliki do piaskownic uruchomieniowych lub innych narzędzi do badania plików w celu dalszego dochodzenia. Zeek ma pewne możliwości wykonywania klasycznego, bajtowo zorientowanego wykrywania włamań, ale to zadanie najlepiej realizują pakiety takie jak otwartoźródłowe silniki Snort lub Suricata. Zeek ma jednak inne możliwości, które potrafią wydawać oceny w postaci alertów, dzięki mechanizmowi notice. Zeek nie jest zoptymalizowany pod kątem zapisywania ruchu na dysk w duchu zbierania pełnej treści, a to zadanie najlepiej realizuje oprogramowanie napisane specjalnie w tym celu. Poza formami danych sieciowych, które Zeek potrafi natywnie zbierać i generować, Zeek ma zalety, które pojawiły się w sekcji Co to jest Zeek? Obejmują one wbudowaną funkcjonalność dla szeregu zadań analizy i detekcji oraz fakt, że jest to w pełni konfigurowalna i rozszerzalna platforma do analizy ruchu. Zeek jest też atrakcyjny ze względu na możliwość działania na powszechnie dostępnych podzespołach, co daje użytkownikom każdego typu możliwość przynajmniej wypyrowowania Zeeka w sposób niskokosztowy. Historia Zeek ma bogatą historię sięgającą lat 90. Vern Paxson zaprojektował i zaimplementował początkową wersję w 1995 roku jako badacz w Lawrence Berkeley National Laboratory, w skrócie LBNL. Oryginalne oprogramowanie nosiło nazwę Bro, jako orwellowskie przypomnienie, że monitorowanie idzie w parze z możliwością naruszeń prywatności. LBNL po raz pierwszy wdrożyło Zeeka w 1996 roku, a USENIX Security Symposium opublikowało oryginalną pracę Verna na temat Zeeka w 1998 roku i przyznało jej w tym roku nagrodę Best Paper Award. W 1999 roku opublikował dopracowaną wersję artykułu jako Bro: A System for Detecting Network Intruders in Real-Time. W 2003 roku National Science Foundation, czyli NSF, zaczęła wspierać badania oraz zaawansowany rozwój Bro w International Computer Science Institute, w skrócie ICSI. Vern nadal kieruje grupą ICSI Networking and Security. Na przestrzeni lat rosnący zespół badaczy i studentów ICSI dodawał do Zeeka nowe funkcje, podczas gdy LBNL kontynuowało wsparcie dzięki finansowaniu z Department of Energy, czyli DOE. Duża część możliwości Zeeka wywodzi się z akademickich projektów badawczych, a wyniki często publikowano na konferencjach najwyższej rangi. Kluczem do sukcesu Zeeka była zdolność projektu do łączenia świata akademickiego z operacjami. Ta relacja pomogła ugruntować badania nad Zeekiem w realnych wyzwaniach. Wraz z rosnącą operacyjną społecznością użytkowników, model rozwoju skoncentrowany na badaniach ostatecznie stał się wąskim gardłem ewolucji systemu. Granty badawcze nie wspierały bardziej przyziemnych części rozwoju i utrzymania oprogramowania. Te elementy były jednak kluczowe dla doświadczenia użytkownika końcowego. W rezultacie wdrożenie Zeeka wymagało pokonania stromej krzywej uczenia się. W 2010 roku NSF próbowała rozwiązać to wyzwanie, przyznając ICSI grant z funduszu Software Development for Cyberinfrastructure. National Center for Supercomputing Applications, czyli NCSA, dołączyło do zespołu jako kluczowy partner, a projekt Zeek rozpoczął przebudowę wielu elementów systemu widocznych dla użytkownika na potrzeby wydania 2.0 w 2012 roku. Po Zeek 2.0 projekt doświadczył ogromnego wzrostu liczby nowych wdrożeń w zróżnicowanych środowiskach, a trwająca współpraca pomiędzy ICSI, współgłówny badacz Robin Sommer, i NCSA, współgłówny badacz Adam Slagell, przyniosła szereg istotnych funkcji. W 2012 roku Zeek dodał natywną obsługę IPv6 na długo przed wieloma narzędziami do monitorowania sieci w środowiskach enterprise. W 2013 roku NSF odnowiła wsparcie drugim grantem, który ustanowił Bro Center of Expertise w ICSI i NCSA, promując Zeek jako kompleksową, niskokosztową zdolność bezpieczeństwa dla społeczności badawczych i edukacyjnych. Aby ułatwić zarówno debugowanie, jak i edukację, w 2014 roku uruchomiono try.zeek.org, wcześniej try.bro.org. Zapewniało to interaktywny sposób testowania skryptu z własnymi przechwyceniami pakietów względem różnych wersji Zeeka oraz łatwe dzielenie się przykładowym kodem z innymi. Dla klastrów Zeeka i komunikacji zewnętrznej dodano framework komunikacyjny Broker. Na koniec, ale nie mniej ważne, w 2016 roku stworzono menedżera pakietów Zeeka, sfinansowanego dodatkowym grantem Mozilla Foundation. Jesienią 2018 roku zespół kierujący projektem zdecydował o zmianie nazwy oprogramowania z Bro na Zeek. Zespół chciał nazwy, która lepiej odzwierciedlałaby wartości społeczności, a jednocześnie unikała negatywnych skojarzeń z tak zwaną kulturą bro poza światem informatyki. Projekt wydał wersję 3.0 jesienią 2019 roku, było to pierwsze wydanie pod nazwą Zeek. Rok 2020 przyniósł ponowne skupienie na społeczności i rozwijaniu społeczności Zeeka, ze zwiększoną aktywnością poprzez media społecznościowe, webinarium, kanały Slack oraz powiązane działania informacyjne. Aby poznać historię projektu od 1995 do 2015 roku, zobacz

wystąpienie Verna Paxsona z BroCon 2015 pod tytułem Reflecting on Twenty Years of Bro. Aby poznać tło decyzji o zmianie nazwy z Bro na Zeek, zobacz wystąpienie Verna Paxsona z BroCon 2018 pod tytułem Renaming Bro.



Na bardzo wysokim poziomie Zeek jest architektonicznie podzielony warstwowo na dwa główne komponenty. Jego silnik zdarzeń, czyli rdzeń, redukuje przychodzący strumień pakietów do serii zdarzeń wyższego poziomu. Zdarzenia te odzwierciedlają aktywność sieciową w kategoriach neutralnych względem polityk, czyli opisują to, co zostało zaobserwowane, ale nie dlaczego ani czy jest to istotne. Na przykład każde żądanie HTTP na łączu zamienia się w odpowiadające mu zdarzenie `http_request`, które niesie ze sobą zaangażowane adresy IP i porty, żądany URI oraz używaną wersję HTTP. Zdarzenie nie przekazuje jednak dalszej interpretacji, na przykład tego, czy dany URI odpowiada znanej stronie z malware. Komponent silnika zdarzeń składa się z szeregu podkomponentów, w tym w szczególności z potoku przetwarzania pakietów obejmującego źródła wejściowe, analizę pakietów, analizę sesji oraz analizę plików. Źródła wejściowe pobierają napływający ruch sieciowy z interfejsów sieciowych. Analiza pakietów przetwarza protokoły niższego poziomu, zaczynając już od warstwy łącza. Analiza sesji obsługuje protokoły warstwy aplikacji, takie jak HTTP, FTP i inne. Analiza plików rozkłada na czynniki zawartość plików przesyłanych w ramach sesji. Silnik zdarzeń udostępnia architekturę wtyczek, która pozwala dodawać dowolny z tych elementów spoza rdzenia bazy kodu Zeeka, umożliwiając rozszerzanie możliwości Zeeka w razie potrzeby. Semantyka związana ze zdarzeniami jest wyprowadzana przez drugi główny komponent Zeeka, interpreter skryptów, który wykonuje zestaw handlerów zdarzeń napisanych w niestandardowym języku skryptowym Zeeka. Skrypty te mogą wyrażać politykę bezpieczeństwa danego miejsca, na przykład jakie działania podjąć, gdy monitor wykryje różne typy aktywności. Bardziej ogólnie skrypty mogą wyprowadzać dowolne pożądane właściwości i statystyki na podstawie ruchu wejściowego. W istocie cały domyślny wynik Zeeka pochodzi ze skryptów dołączonych do dystrybucji. Język Zeeka zawiera rozbudowane, dziedzinowe typy oraz funkcje pomocnicze. Kluczowe jest to, że język Zeeka pozwala skryptom utrzymywać stan w czasie, co umożliwia śledzenie i korelowanie ewolucji obserwacji ponad granicami połączeń i hostów. Skrypty Zeeka mogą generować alerty w czasie rzeczywistym oraz na żądanie uruchamiać dowolne zewnętrzne programy. Można użyć tej funkcjonalności do uruchomienia aktywnej reakcji na atak.

Zadania praktyczne

Zadanie 1 – LetsDefend i Tryhackme

Wykonaj lab: <https://tryhackme.com/room/azuresentinel-aoc2025-a7d3h9k0p2>

Wykonaj lab: <https://app.letsdefend.io/training/lessons/incident-management-101>

Laby dodatkowe:

LetsDefend:

- <https://app.letsdefend.io/training/lessons/phishing-email-analysis>
- <https://app.letsdefend.io/training/lessons/web-attacks-101>
- <https://app.letsdefend.io/challenge/investigate-web-attack>
- <https://app.letsdefend.io/training/lessons/security-solutions>
- <https://app.letsdefend.io/training/lessons/network-log-analysis>

Tryhackme:

- <https://tryhackme.com/room/introtoendpointsecurity>
- <https://tryhackme.com/room/idsevasion>
- <https://tryhackme.com/room/threatintelligenceforsoc>

Zadanie 2 – Snort

- <https://www.snort.org/>
- <https://letsdefend.io/blog/how-to-install-and-configure-snort-on-ubuntu>

Uruchom lab: <https://tryhackme.com/room/snort>

Alternatywnie Konfiguracja manualna:

- <https://hackertarget.com/snort-tutorial-practical-examples/>
- <https://www.youtube.com/watch?v=UAKq1S-VxJ8>
- <https://medium.com/@hammazahmed40/snort-a-step-by-step-guide-to-writing-and-testing-simple-rules-0914094b1b7b>

Instalacja Snort na Ubuntu

<https://letsdefend.io/blog/how-to-install-and-configure-snort-on-ubuntu>

```
sudo apt update && sudo apt upgrade -y  
sudo apt install snort -y
```

jak nie działa to wtedy można wykorzystać docker:

```
sudo docker pull ciscotalos/snort3  
sudo docker run --name snort3 -h snort3 -u snorty -w /home/snorty -d -it ciscotalos/snort3 bash  
sudo docker exec -it snort3 bash
```

Podczas instalacji:

- interfejs sieciowy => wybierz np. eth0
- sieć domyślna => np. 192.168.0.0/24

Podstawowa konfiguracja Snorta

- **Główny plik konfiguracyjny:**

/etc/snort/snort.conf

Lokalizacja reguł:

/etc/snort/rules/local.rules => Tu będziemy dodawać własne reguły

Uruchomienie Snorta

```
sudo snort -A console -q -c /etc/snort/snort.conf -i eth0
```

Co oznaczają flagi?

- -A console => alerty na ekran (może nie działać w nowszej wersji snort)
- -q => tryb cichy
- -c => plik konfiguracyjny
- -i => interfejs sieciowy

Proste przykłady reguł Snorta

Przykład 1 – ICMP (ping)

Dodaj do local.rules:

```
alert icmp any any -> any any (msg:"ICMP Ping detected"; sid:1000001; rev:1;)
```

Test:

```
ping google.com
```

REGUŁA: HTTP na niestandardowym porcie (wymagane w zadaniu)

Wykryć próbę użycia HTTP na porcie innym niż 80

Reguła Snorta:

```
alert tcp any any -> any !80 (
    msg:"HTTP traffic on non-standard port";
    flow:to_server,established;
    content:"GET";
    sid:1000002;
    rev:1;
)
```

Test:

```
curl http://example.com:8080
```

Zadanie 1 – Ping (Tak jak w przykładzie 1)

1. Dodaj regułę wykrywającą ICMP
2. Wykonaj ping
3. Sprawdź alert

Zadanie 2 – HTTP na niestandardowym porcie (Tak jak przykładzie 2)

1. Dodaj regułę wykrywającą HTTP na porcie ≠ 80
2. Wykonaj curl na porcie 8080
3. Opisz alert

Zadanie 3 – Telnet (niebezpieczny protokół)

Dodaj regułę:

```
alert tcp any any -> any 23 (msg:"TELNET connection detected"; sid:1000003; rev:1;)
```

Test:

telnet example.com

Zadanie 4 – Prosty skan portów

Dodaj regułę:

```
alert tcp any any -> any any (flags:S; msg:"Possible port scan"; sid:1000004; rev:1;)
```

Test:

nmap localhost

Zadanie 3 – Suricata

- <https://suricata.io/>
- <https://medium.com/@redfanatic7/suricata-vs-snort-detailed-guide-to-the-programs-c331cff452a1>

Zainstaluj i uruchom suricate wykorzystując poradnik:

- <https://letsdefend.io/blog/how-to-install-and-configure-suricata-on-ubuntu>
- <https://docs.suricata.io/en/latest/quickstart.html>

Podstawowa konfiguracja Suricaty

Główny plik konfiguracyjny:

/etc/suricata/suricata.yaml

Plik reguł lokalnych:

/etc/suricata/rules/local.rules

Przykłady:

- <https://github.com/0xrajneesh/Suricata-IDS-Home-Lab>

Przykład 1 – ICMP (ping)

Przeczytaj: <https://docs.suricata.io/en/latest/rule-management/adding-your-own-rules.html>

Następnie:

Dodaj do local.rules:

```
alert icmp any any -> any any (msg:"ICMP Ping detected - Suricata"; sid:2000001; rev:1;)
```

Test:

ping google.com

REGUŁA: HTTP na niestandardowym porcie

Cel:

Wykrycie HTTP na porcie innym niż 80

Reguła Suricaty:

```
alert http any any -> any !80 (
    msg:"HTTP traffic on non-standard port (Suricata)";
    sid:2000002;
    rev:1;
)
```

Test:

curl http://example.com:8080

Sprawdzenie alertu:

sudo tail -f /var/log/suricata/fast.log

Zadanie 1 – Ping (Tak jak w przykładzie 1)

1. Dodaj regułę ICMP
2. Wykonaj ping
3. Sprawdź fast.log

Zadanie 2 – HTTP na niestandardowym porcie (Tak jak w przykładzie 2)

1. Dodaj regułę HTTP
2. Wykonaj curl :8080
3. Opisz alert

Zadanie 3 – DNS

Dodaj regułę:

```
alert dns any any -> any any (
    msg:"DNS query detected";
    sid:2000003;
    rev:1;
)
```

Test:

```
dig google.com
```

Zadanie 4 – Telnet

```
alert tcp any any -> any 23 (
    msg:"TELNET connection detected (Suricata)";
    sid:2000004;
    rev:1;
)
```

Zadanie 5 – Skan portów

```
alert tcp any any -> any any (
    flags:S;
    msg:"Possible port scan detected (Suricata)";
    sid:2000005;
    rev:1;
)
```

Test:

```
nmap localhost
```

Zadanie 4 – Zeek

- <https://github.com/zeek/zeek>
- <https://letsdefend.io/blog/how-to-install-zeek-on-linux>

instalacja Zeeka

<https://docs.zeek.org/en/master/install.html>

```
echo 'deb https://download.opensuse.org/repositories/security:/zeek/xUbuntu_22.04/' | sudo tee  
/etc/apt/sources.list.d/security:zeek.list  
curl -fsSL https://download.opensuse.org/repositories/security:zeek/xUbuntu_22.04/Release.key | gpg --dearmor |  
sudo tee /etc/apt/trusted.gpg.d/security_zeek.gpg > /dev/null  
sudo apt update  
sudo apt install zeek-7.0
```

Podstawowa konfiguracja Zeeka

Pliki konfiguracyjne:

```
/etc/zeek/  
Najważniejsze:  
• node.cfg  
• networks.cfg  
• zeekctl.cfg
```

Analiza ruchu – podstawowe logi Zeeka

Logi zapisują się w:

```
/opt/zeek/logs/current/
```

Proste przykłady

<https://docs.zeek.org/en/master/quickstart.html>

http

zeek -i eth0 -C

```
curl http://example.com  
Sprawdź:  
cat /opt/zeek/logs/current/http.log
```

DNS

```
dig google.com  
cat /opt/zeek/logs/current/dns.log
```

SSH

```
ssh localhost  
cat /opt/zeek/logs/current/ssh.log
```

Zadanie 1 – Analiza połączeń

1. Wykonaj kilka zapytań HTTP i DNS
2. Otwórz conn.log
3. Odpowiedz:
 - kto łączył się z kim?
 - jaki protokół?
 - jaki port?

Zadanie 2 – HTTP na niestandardowym porcie

```
curl http://example.com:8080  
cat /opt/zeek/logs/current/http.log
```

Zeek pokazuje HTTP mimo innego portu

Zadanie 3 – DNS tunneling (symulacja)

```
dig test.example.com  
Sprawdź:  
cat /opt/zeek/logs/current/dns.log
```

Czy domena wygląda normalnie?

Zadanie 4 – Nietypowy port SSH

```
ssh -p 2222 localhost  
Sprawdź:  
cat /opt/zeek/logs/current/ssh.log
```

Prosty alert w Zeeku (skrypt)

Zeek używa skryptów, nie reguł.

Skrypt: HTTP na porcie ≠ 80

Plik:

/opt/zeek/share/zeek/site/http_nonstandard.zeek

```
event http_request(c: connection, method: string, host: string, uri: string)  
{  
    if ( c$id$resp_p != 80 )  
        print fmt("HTTP on non-standard port: %s:%s",  
                 c$id$resp_h, c$id$resp_p);  
}
```

Dodaj do:

- /opt/zeek/share/zeek/site/local.zeek

```
@load ./http_nonstandard.zeek
```

Restart:

```
sudo zeekctl deploy
```

Zadanie 5 – Dodatkowe – Dla chętnych

Spróbować lokalnie skonfigurować rozwiązania WAF:

- <https://github.com/owasp-modsecurity/ModSecurity>
- <https://github.com/chaitin/SafeLine>

Zadanie 6 - dodatkowe – dla chętnych 😊

Budowa lokalnego Lab

Spróbować zbudować własny mini lab 1 maszyna Windows 1 Linux + zbieranie logów.

Można wykorzystać:

- <https://github.com/nakairuzive/How-to-Build-Your-First-SOC-Home-Lab-Beginner-s-Guide->
- <https://github.com/gavinpaultech/SOC-Lab>
- <https://github.com/uruc/SOC-Automation-Lab>
- <https://github.com/kyhomelab/SOC-Lab>
- <https://github.com/xAHINX00/SOC-Home-Lab-Attack-Defense-Simulation>
- <https://github.com/praiseordu/SOC-ANALYSIS-LAB>
- https://github.com/hyphennR/Document_SOC_Home_Lab
- <https://medium.com/@nakkouchtarek/from-zero-to-soc-homelab-my-journey-to-defense-in-depth-and-full-security-automation-cc07373b8b6d>
- <https://app.letsdefend.io/training/lessons/building-a-soc-lab-at-home>