System detekcji skanowania portów metodą analizy statystycznej

Jakub Kowalczuk
Wydział Elektroniki i Technik Informacyjnych
Politechnika Warszawska
Warszawa, Polska
jakub.kowalczuk.stud@pw.edu.pl

Aleksandra Brzeszczyńska
Wydział Elektroniki i Technik Informacyjnych
Politechnika Warszawska
Warszawa, Polska
aleksandra.brzeszczynska.stud@pw.edu.pl

***Streszczenie*—W niniejszej pracy dokonano opisu oraz testów wykonanej implementacji systemu opartego na algorytmie TRW, pozwalającego na detekcje ataków skanujących porty z wykorzystaniem protokółu TCP. System przetestowano na zbiorze CICIDS2017 oraz pod kątem różnych scenariuszy ataków. Zbadano reakcje systemu na różne stopnie nasilenia połączeń zakończonych porażką oraz sprawdzono jego wydajność działania. Zaprojektowany system pozwala uzyskać dobrą skuteczność detekcji dla skanów horyzontalnych jak i wertykalnych w atakach z jednego źródła.**

***Słowa kluczowe*—skanowanie portów, system detekcji, atak sieciowy, TRW, rekonesans, TCP**

I. Wstęp

Przeprowadzenia skutecznego i efektywnego ataku sieciowego wymaga wieloetapowego postępowania: od zdobycia jak najszerszej wiedzy o topologii sieci ofiary, po wydobycie pożądanych informacji lub dokonanie różnego rodzaju akcji negatywnie wpływających na prawidłowe działanie sytemu. W praktyce wyróżniamy pięć typowych faz ataku: rekonesans, wykorzystanie słabości, zwiększenie kontroli nad ofiarą, instalowanie tylnej furtki oraz plądrowanie. W ramach niniejszej pracy skupimy się na pierwszej fazie jaką jest rekonesans, a konkretnie skanowaniem portów. Celem tego działania jest precyzyjne określenie luk w serwerach i zdiagnozowanie poziomu zabezpieczeń. Skanowanie odbywa się poprzez wysłanie sekwencji pakietów sondujących do docelowych portów sieciowych i obserwację ich odpowiedzi. Otwarte porty bez odpowiednich zabezpieczeń stanowią potencjalną furtkę wejściową dla atakującego do sieci ofiary. Po wykryciu słabości związanych z usługami skojarzonymi z danym portem intruz, jest w stanie przeprowadzić atak. Ponieważ skanowanie portów jest jednym z najczęstszych prekursorów włamań do sieci, ich wykrycie jest kluczowe do zapewnienia maksymalnego poziomu bezpieczeństwa sieci. W przypadku detekcji skanowania ważna jest również szybkość stwierdzenia, że seria połączeń stanowi wrogą aktywność. Termin „szybkość" odnosi się tutaj do liczby kolejnych działań atakującego (liczby przeskanowanych hostów/portów), a naszym celem jest wykrycie go we wczesnej fazie, zanim zdobędzie zbyt wiele informacji.

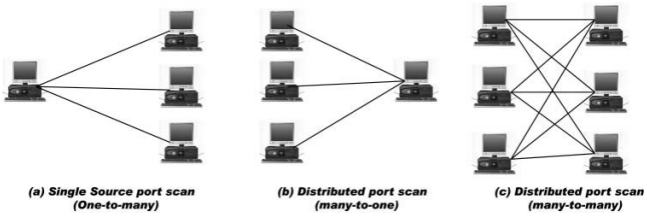
A. Typy skanowania

Możemy wyróżnić trzy typowe scenariusze skanowania portów [15]:

- Jeden do wielu (*ang."one-to-many"*) Atak z jednego źródła przeprowadzony na sieć ofiary (Rys. 1(a)).
- Wiele do jednego (*ang. "one-to-many"*) Wiele hostów atakującego z których każdy skanuję jedno urządzenie ofiary (Rys. 1(b)).
- Wiele do wielu (*ang. "many-to-many"*) Wiele hostów atakującego z skanujących wile urządzenie w sieci ofiary (Rys. 1(c)).

Należy zauważyć, że scenariusze wykorzystujące wiele hostów przez atakujących są trudniejsze do wykrycia, natomiast wymagają większych zasobów w postaci dostępnych adresów IP (*ang."Internet Protocol address"*). Ponieważ pula adresacji IP dla wersji czwartej (IPv4) została wyczerpana, a adresacja w wersji szóstej(IPv6) wciąż nie jest globalnie stosowana (szczególnie w Europie), zdobycie takich zasobów może stanowić wyzwanie.
W dalszej części pracy skupimy się na atakach z pojedynczego źródła, te możemy podzielić na skanowania horyzontalne i wertykalne. Pierwsze z nich dokonują skanu pojedynczych portów na wielu maszynach w sieci ofiary, natomiast drugie dokonują skanowania wielu portów pojedynczego hosta.

Kolejnym kryterium podziału metod skanowania portów jest prędkość. Atakujący może dokonać skanowania wybranych przez niego portów w różnych odstępach czasu. W przypadku szybkiego skanowania pełny skan może być kwestią sekund, natomiast w przypadku wolnego, skan może trwać miesiącami. Z tego powodu istotne jest, aby systemy detekcji potrafiły skupić się zarówno na bieżącym zapisie ruchu jak i długoterminowej historii połączeń.



Rysunek 1. Typowe scenariusze skanowania portów. **(a)** Jeden do wielu. **(b)** Wiele do jednego. **(c)** Wiele do wielu. [15]

B. Wykonana implementacja systemu

W niniejszej pracy zaimplementowano rozwiązanie detekcji z wykorzystaniem metodo analizy statystycznej, a następnie przeprowadzono badania w celu stwierdzenia jego skuteczności. Bazowa metoda stanowi dokładne odwzorowanie algorytmu TRW(*ang. Threshold Random Walk*) zaproponowanej w artykule *"Fast Portscan Detection Using Sequential Hypothesis Testing"* [12]. Rozwiązanie zakłada wykrycie ataków przeprowadzonych z pojedynczego źródła na wiele hostów w sieci ofiary z wykorzystaniem protokółu TCP (*ang. "Transmission Control Protocol"*) [6]. Po analizie działania algorytmu TRW zdecydowaliśmy się na jego modyfikację, która działa równolegle z wersją oryginalną. Wprowadzone zmiany umożliwiają wykrywanie skanowania wertykalnego na pojedynczym hostcie ofiary. Uzyskany system, w wielu scenariuszach, jest w stanie wykryć skaner po zaledwie czterech próbach połączenia, dla obu wariantów.

C. Organizacja pracy

Praca została podzielona na następujące rozdziały. W Rozdziale II dokonano przeglądu literatury dotyczącej wykorzystywanego algorytmu. Rozdział III jest teoretycznym wprowadzeniem zaimplementowanego rozwiązania, następnie Rozdział IV opisuje jego implementacje. W Rozdziałach V, VI oraz VII omówione zostały kolejno wykonane eksperymenty, ich wyniki, oraz testy wydajnościowe systemu.

II. Przegląd literatury

Istnieje wiele metod wykrywania skanowania portów projektowanych pod kątem różnego rodzaju ataków. Artykuł [8] wyróżnia następujące metody systemów detekcji ataków typu *ang. one to many*:

- metody algorytmiczne (*ang. alghoritmic aproaches*) oparte najczęściej na testowaniu hipotez i metodach probabilistycznych,
- metody oparte na progach (*ang. threshold based*), które na podstawie zaobserwowanych połączeń wyznaczają pewną wielkość (często liczbę unikatowych hostów z którymi komunikował się potencjalny atakujący), a następnie konfrontują ją z ustalonym progiem.
- metody oparte o wzorce (*ang. rule based approaches*, które na podstawie zaobserwowanej historii ruchu sieciowego, generują wzorce spodziewanych zachowań i wychwytują ruch od nich odbiegający.

Wykorzystywany w niniejszej pracy algorytm Threshold Random Walk (TRW) jest metodą opartą na progach, szeroko cytowaną w literaturze tematycznej. Nie jest pozbawiony wad, co wykazane zostało między innymi w pracach [13] czy [14], wskazujących na brak odporności algorytmu na techniki stosowane przez atakujących mających świadomość jego wykorzystywania. Mimo tego, ze względu na dużą skuteczność potwierdzoną przez autorów prac [12] oraz [10] algorytm TRW stanowi podstawę wielu stosowanych rozwiązań opartych na jego modyfikacjach [11], [13], takich jak na przykład (co opisano w artykule [14]) rozszerzenie działania algorytmu na ruch oparty na protokołach innych niż TCP.

III. Metoda Threshold Random Walk [12]

Jedną z głównych cech skanerów jest to, że częściej niż legalne zdalne hosty wybierają hosty, które nie istnieją lub nie mają aktywowanej żądanej usługi. Wynika to z faktu, że nie mają wiedzy na temat tego, które hosty i porty w sieci docelowej są obecnie aktywne. Autorzy metody w sowich badaniach zauważyli fakt iż dla niegroźnych hostów prawidłowe połączenia stanowią ponad 80% ich ruchu. Na podstawie tej obserwacji sformułowano problem, a następnie algorytm detekcji Threshold Random Walk (TRW), którego celem jest oflagowanie złośliwej aktywności.

A. Model

Zdarzenie zostaje wygenerowane, gdy zdalne źródło r podejmie próbę połączenia z lokalnym miejscem docelowym l. Wynik zdarzenia klasyfikujemy binarnie jako "sukces"lub "porażkę". Przy czym "porażką"określamy próbę połączenia z nieaktywnym hostem lub nieaktywną usługą w aktywnym hoście.

Dla danego r, Y_i jest zmienną losową (wskaźnikową), reprezentującą wynik pierwszej próby połączenia r z $i-tym$ unikalnym hostem lokalnym, gdzie

$$Y_i = \begin{cases} 1 & \text{jeśli połączenie zakończyło się sukcesem} \\ 0 & \text{w p.p.} \end{cases} \tag{1}$$

Obserwując wyniki $Y_1, Y_2, ...$ określamy czy r to skaner.

B. Sekwencyjne testowanie hipotez

Rozważamy dwie hipotezy H_0 i H_1, gdzie H_0 jest hipotezą, że dane źródło r jest nieszkodliwe, natomiast H_1 zakłada, że r to skaner.

Dodatkowo zakładamy, że w zależności od hipotezy H_j zmienne losowe $Y_i|H_j$ $i = 1, 2, ...$ są niezależne i mają identyczny rozkład. Wtedy wyrażamy rozkład zmiennej losowej Y_i Bernoulliego jako:

$$\begin{aligned} Pr[Y_i = 0|H_0] = \theta_0, Pr[Y_i = 1|H_0] = 1 - \theta_0 \\ Pr[Y_i = 0|H_1] = \theta_1, Pr[Y_i = 1|H_1] = 1 - \theta_1 \end{aligned} \tag{2}$$

Obserwacja, że próba połączenia ma większe szanse na powodzenie z nieszkodliwego źródła niż złośliwego implikuje warunek

$$\theta_0 > \theta_1. \tag{3}$$

Dodatkowym wykorzystywanymi parametrami określającymi warunki działania algorytmu detekcji jest prawdopodobieństwo detekcji P_D i prawdopodobieństwo fałszywej detekcji P_F. Dla wybranych przez użytkownika wartości α i β oczekujemy spełnienia zależności:

$$P_F \leq \alpha \text{ oraz } P_D \geq \beta, \tag{4}$$

gdzie typowa wartość $\alpha = 0.01$, a $\beta = 0.99$.

Celem algorytmu wykrywania w czasie rzeczywistym jest podjęcie wczesnej decyzji, kiedy strumień zdarzeń dociera do

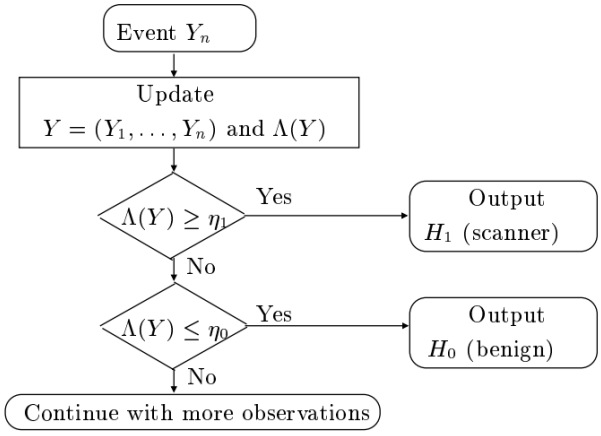
systemu. W miarę obserwowania każdego zdarzenia określamy iloraz prawdopodobieństw:

$$\Lambda(Y) = \frac{Pr[Y|H_1]}{Pr[Y|H_0]} = \Pi_{i=1}^{n} \frac{Pr[Y_i|H_1]}{Pr[Y_i|H_0]} \tag{5}$$

gdzie Y to wektor zaobserwowanych dotychczas zdarzeń, a $Pr[Y|H_i]$ reprezentuje funkcję masy prawdopodobieństwa warunkowego strumienia zdarzeń Y, przy założeniu, że zdarzenia Y_i są niezależne. Następnie porównujemy iloraz prawdopodobieństw z górnym progiem (*ang. upper threshold*) η_1 i dolnym progiem (*ang. lower threshold*) η_0:

- Jeśli $\Lambda(Y) \leq \eta_0$ to przyjmujemy hipotezę zerową H_0.
- Jeśli $\Lambda(Y) \geq \eta_1$ to przyjmujemy hipotezę alternatywną H_1.
- Jeśli $\eta_0 < \Lambda(Y) < \eta_1$ to czekamy na następną obserwację i aktualizujemy $\Lambda(Y)$.

Całość algorytmu przedstawia schemat blokowy Rysunek 2 z artykułu źródłowego [12].



Rysunek 2. Schemat blokowy algorytmu TRW. [12]

Progi powinny być dobrane tak, aby spełnione były oba warunki prawdopodobieństwa z równania (4). Wartości progów nie są z góry jasne, jednak kluczową cechą sekwencyjnego testowania hipotez jest możliwość określenia ich poprzez proste wyrażenia z użyciem α i β. Trzeba również pamiętać, że wartości progów mają wpływ na N liczbę obserwacji do zakończenia testu — momentu wybrania jednej z hipotez.

Ponadto $\eta_0(\eta_1)$ może być ograniczona z góry (z dołu) prostymi wyrażeniami P_F i P_D. Te wyrażenia mogę również posłużyć jako praktyczne przybliżenie progów, gdzie P_F i P_D są zastępowane wybranymi przez użytkownika α i β. Alternatywnie użytkownik może sam wybrać progi η_0 oraz η_1 znając zależności pomiędzy prawdopodobieństwami P_D oraz P_F, a α i β.

C. Ograniczenia

W zależności od hipotezy (czy zdalny host jest skanerem czy nie), dowolne dwie próby połączenia będą miały takie samo prawdopodobieństwo powodzenia, a ich szanse powodzenia nie są ze sobą powiązane. Górna granica progu η_1 definiujemy następująco:

$$\eta_1 \leq \frac{P_D}{P_F} \tag{6}$$

Analogicznie dolna granica η_0:

$$\frac{1 - P_D}{1 - P_F} \leq \eta_0 \tag{7}$$

Problematyczną w kontekście wybranej metody sytuacją może być gdy skaner sonduje N nieaktywnych serwerów dokładnie na przemian z N serwerami aktywnymi. W takim przypadku wartość ilorazu prawdopodobieństwa oscylować będzie miedzy dolnym i górnym progiem, bez możliwości ostatecznego przypisania klasy.

IV. Implementacja

Do implementacji wykorzystano język Python [3] w wersji 3.11 oraz bibliotekę Scapy [5] w wersji 2.5.0 udostępniającą funkcjonalności pozwalające na przechwytywanie i analizę ruchu sieciowego.
System podzielono na dwa główne moduły: nasłuchujący ruch (*ang. Sniffer*) oraz moduł analizujący metodą TRW. Pakiety ze Sniffera filtrowane są wstępnie w celu wydobycia jedynie ruchu TCP, a następnie przekazywane do analizy. Analiza przeprowadzona jest w oddzielnym wątku. Do synchronizacji danych wykorzystano obiekt kolejki (Queue) natywnie wspieranej przez język Python. Pełen kod aplikacji można znaleźć na repozytorium w serwisie GitHub [4].

A. Wyrocznia

W celu podjęcia decyzji czy dane połączenie TCP się powiodło analogicznie do autorów metody TRW zaimplementowano "wyrocznię"(*ang. "Orcale"*). Wiedza przechowywana przez wyrocznię zawiera listę wszystkich otwartych portów na hostach z chronionej sieci. Każde pakiet inicjalizujący połączenie TCP (flaga SYN) do hosta z wewnątrz sieci ofiary jest konfrontowany z listą: jeżeli wpis w postaci *docelowe_IP:docelowy_port* istnieje na liście połączenie uznaję się jako sukces, w przeciwnym wypadku jako porażkę. Takie podejście wprowadza konieczność utrzymywania aktualnego stanu wiedzy, jednak ma swoje dodatkowe zalety takie jak: oznaczanie połączeń do przypadkowo pozostawionych otwartych portów jako porażka oraz brak zaburzania statystyk dla prawidłowych połączeń przy potencjalnej awarii sieci.

B. Modyfikacja metody TRW

W implementacji zdecydowano się na rozszerzenie metody do wykrywania skanów wertykalnych. W tym celu dokonano drobnej zmiany w module TRW, który tworząc listę unikalnych połączeń potencjalnego atakującego jako klucz przyjmuje parę IP i port hosta docelowego. Zmodyfikowany moduł (TRWP) działa równolegle z oryginalnym dokonując niezależnych detekcji.

V. Eksperymenty

A. Zbiór danych

1) Opis zbioru: W procesie projektowania narzędzi typu NIDS, kluczowy jest wybór odpowiedniego zbioru danych, w celu analizy i testowania proponowanych rozwiązań. Zbiór danych CICIDS2017 [9] utworzono na podstawie zapisu symulacji ruchu sieciowego zebranego podczas ataków różnego typu t.j: Brute Force FTP, Brute Force SSH, DoS, Heartbleed, Web Attack, Infiltration, Botnet, DDoS oraz PortScan. Zbiór dotyczący skanowania portów utworzony jest z przefiltrowanego ruchu TCP oraz UDP(*ang. "User Datagram Protocol"*) i występuje w dwóch wersjach:

- MachineLearningCSV - przetworzony ruch przez narzędzie CICFlowMeter [1] nadający się do uczenia algorytmów uczenia maszynowego
- GeneratedLabelledFlows - wybrane pakiety z parametrami o połączeniu (adresy ip, porty, flagi tcp, itp.)

W obu zbiorach każdy wpis oznaczono jako BENIGN lub PortScan. Wstępna analiza danych wykazała problemy i nieścisłości w zbiorze: pominięcie pakietów inicjalizujących połączenie TCP (flaga SYN) czy niepoprawne ich oznaczenie. Problem z jakością zbioru CICIDS2017 jest znany i opisywany między innymi w artykule [16]. Z tego względu zdecydowano się na wykorzystanie pełnego zapisu ruchu, na podstawie którego został on stworzony(udostępniony w formacie *.pcap*), i samodzielną jego obróbkę.

2) Obróbka danych: Do dalszej analizy wybrano fragment zapisu ruchu dla CICIDS2017 zebrany w Piątek, 7.06.2017 w godzinach popołudniowych, zawierający dane dotyczące skanowania portów (*ang. port scan* przez jednego atakującego(adres IP: 172.16.0.1). Analizę i filtrowanie danych przeprowadzono przy pomocy narzędzia Wireshark [7]. Zaproponowane rozwiązanie dotyczy analizy ruchu TCP, wykonano więc filtracje wyodrębniającą jedynie pakiety tego protokołu. W celu stworzenia listy wiedzy wyroczni, wybrano wszystkie unikatowe adresy IP sieci lokalnej wraz z portami z których zarejestrowano wysłanie pakietu SYN/ACK stanowiącego drugi etap nawiązywania połączenia TCP. Rozwiązanie to obarczone jest pewnym błędem - wszystkie połączenia, które nie wystąpiły w analizowanym zbiorze, uznawane są za nieudane (zakończone porażką), a wszystkie zaobserwowane z odpowiednimi flagami - jako udane, co może nie odzwierciadlać rzeczywistej sytuacji. Lista hostów(adresów IP) z sieci lokalnej pozyskana został na podstawie opisu udostępnionego przez autorów zbioru [9].

B. Platforma testowa

Do przeprowadzenia eksperymentów zaimplementowany system uruchomiono na platformie testowej (**H1**):

- Windows 10
- Intel I5-13600KF
- 32GB RAM (3200Mhz)

Parametry algorytmu TRW(dla obu wariantów) przyjęto zgodnie z optymalnymi wartościami przedstawionymi w oryginalnej implementacji:

- $P_d = 0.01$,
- $P_f = 0.99$,
- $\theta_0 = 0.8$,
- $\theta_1 = 0.2$.

Progi decyzyjne wyniosły odpowiednio z równań 6, 7: $\eta_0 = 0.(01)$ oraz $\eta_1 = 99$

C. Topologia sieci testowej

Sieć testowa składała się z 6 hostów w sieci lokalnej. Symulację ataków za pomocą programu *nmap* dokonywano z:

- hosta **H1** na pozostałe urządzenia w sieci
- hosta **H2** na hosta **H1**

Takie podejście umożliwiło na dogodny dostęp do całego, interesującego z punktu widzenia symulacji, ruchu sieciowego.

Dodatkowo w celu uzyskania statystyk zaimplementowanego systemu wykonano dodatkowe skanowania, ręcznie tworząc pakiety za pomocą biblioteki Scapy używając sztucznych adresów hostów źródłowych.

D. Zbiór danych CICIDS2017

Na podstawie utworzonego zapisu ruchu ze wstępnej obróbki danych na zbiorze CICIDS2017, dokonano jego symulacji przez system. Zachowanie relacji czasowych nie było istotne ponieważ system jest wrażliwy jedynie na kolejność sekwencji pakietów, a nie tępo ich wysyłania.

E. Symulowanie ataków

1) Scenariusze testowe: W celu zbadania poprawności działania systemu detekcji przeprowadzono następujące scenariusze testowe z wykorzystaniem narzędzia nmap [2]. Badano trzy typy skanów - do atakowanych hostów kierowano jedynie pakiet SYN rozpoczynający połączenie TCP, bez ACK (skan typu *ang. "stealth scan"*), jedynie pakiet TCP z flagą ACK (skan typu *ang. ACK scan*) oraz nawiązywano próbę pełnego połączenia TCP.

Przeprowadzono testy o następujących scenariuszach:

(a) Atak hosta **H1** na pozostałe urządzenia w sieci, próba nawiązania pełnego połączenia TCP.
(b) Atak hosta **H2** na hosta **H1**, próba nawiązania pełnego połączenia TCP.
(c) Atak hosta **H1** na pozostałe urządzenia w sieci, skan typu *ang. stealth*.
(d) Atak hosta **H2** na hosta **H1**, skan typu *ang. stealth*.
(e) Atak hosta **H1** na pozostałe urządzenia w sieci, skan typu *ACK*
(f) Atak dwóch hostów **H2** i **H1**, próba nawiązania pełnego połączenia TCP. Na pozostałe urządzenia w sieci, na niektórych uruchomiony był program antywirusowy

F. Ataki o różnym nasileniu

W zaprojektowanym rozwiązaniu kluczowym elementem jest kolejność i stosunek połączeń zakończonych porażką do wszystkich. W celu zbadania wrażliwości systemu na takie zmienne utworzono sztuczną listę wiedzy wyroczni zawierającą listę dostępnych otwartych portów. Następnie generowano

ruch losując w zmiennych proporcjach pakiety z listy (połączenie zakończone sukcesem) oraz z poza niej (połączenie zakończone porażką). Stosunek pakietów nieprawidłowych do wszystkich wynosił odpowiednio: 0, 0.05, 0.1, 0.2, 0.4, 0.5, 0.7, 0.8. Dla każdej wartości losowano 50 pakietów, a próbę powtórzono 10 razy.
W czasie eksperymentu zbierano informacje o historii przypisanego hostowi atakującemu ilorazowi prawdopodobieństw 5 oraz czy został wykryty jako skaner.

VI. Wyniki

A. Badania na zbiorze CICIDS2017

System dokonał prawidłowej detekcji dla obu wariantów analizy. Moduł TRW dokonał wykrycia po 6 wykonanych skanach portów w tym 4 zakończonych porażką, natomiast TRWP dla 12 i 7 zakończonych porażką. W pełnym przebiegu dla TRW zarejestrowano 1004 skany, gdzie 997 się nie powiodło, dla TRWP 6, gdzie 5 się nie powiodło. W obu przypadkach skaner został wykryty prawidłowo, hosty z sieci lokalnej oznaczono jako niewrogie, natomiast inne z uwagi na jedyni pojedyncze połączenia pozostały w stanie nieokreślonym.

Należy zaznaczyć, że ten sam eksperyment wykonany na zbiorze CICIDS2017 GeneratedLabelledFlow nie zakończył się powodzeniem(wszystkie hosty z sieci lokalnej oznaczone jako skanery) co było motywacją do analizy i stworzenia własnego wycinka ruchu zapisanego dla ataku PortScan.

B. Symulowanie ataków

Scenariusze testowe zakończyły się następującymi wynikami:

(a) Zarówno implementacja TRW i TRWP poprawnie zidentyfikowała atakującego po 4 otrzymanych pakietach (3 z nich były nieudanymi połączeniami).
(b) TRW nie wykrył skanera (atakowany był tylko jeden unikatowy adres IP). TRWP wykrył atakującego po 8 pakietach (5 nieudanych połączeń), kilkukrotnie jednak zmieniał dezycje - wartość parametru Ls oscylowała wokół progu wyboru hipotezy. Dłuższy czas reakcji systemu wynikał z tego, że pierwsze kilka prób połączeń zakończonych było sukcesem, co potwierdza wpływ kolejności wysyłanych pakietów na szybkość reakcji systemu.
(c) TRW oraz TRWP wykryły skaner. Zaimplementowane rozwiązanie jest w stanie wykrywać ataki typu *ang. stealth*
(d) TRW ani TRWP nie wykryły skanera. Zaproponowane rozwiązanie nie wykrywa ataków typu ACK.
(e) Antywirus zainstalowany na urządzeniu z uruchomionym systemem detekcji zablokował podejrzany ruch sieciowy, jednak zarówno TRW jak i TRWP wykryły skaner chwilę wcześniej, co wskazuje na dużą szybkość zaproponowanego rozwiązania.

Tabela I
Statystyki szybkości diagnozy hosta atakującego

ratio	min packets	max packets	avg packets
0	50	50	50.0
0.05	50	50	50.0
0.1	50	50	50.0
0.2	50	50	50.0
0.4	4	50	33.4
0.5	4	50	29.2
0.7	4	30	11.6
0.8	4	8	5.8

C. Ataki o różnym nasileniu

Poniższe rysunki Rys.3, Rys. 4, Rys. 5 przedstawiają przebieg średniej wielkości ilorazu prawdopodobieństwa Ls w zależności od ilości przesłanych pakietów, w ramach 10 przeprowadzonych prób dla każdego stosunku (*ang. ratio*) połączeń zakończonych porażką do wszystkich połączeń. Wielkości η_0, η_1 stanowią progi przyjęcia hipotez wyznaczone na podstawie równań 6 i 7.

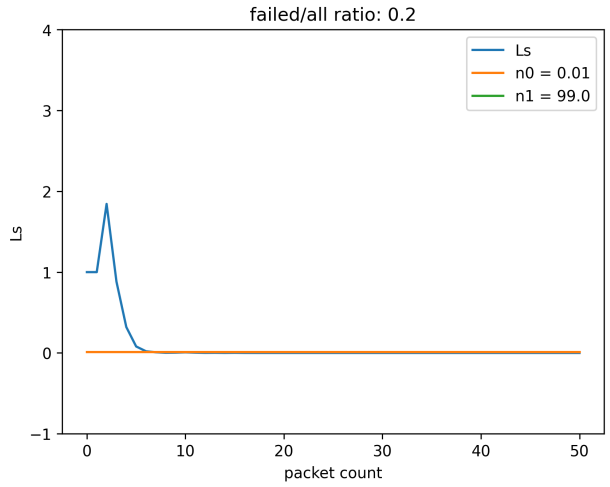
Rysunki 3, 4, 5 ilustrują jak duży wpływ ma stosunek połączeń zakończonych porażką w stosunku do wszystkich wysyłanych połączeń. Przy małej ich ilości (Rys. 3, wartość ilorazu Ls szybko osiąga wartość η_0, co pozwala systemowi detekcji na sklasyfikowanie hosta jako niegroźnego. W przypadku, gdy stosunek jest wyrównany (Rys. 4) wartość Ls oscyluje wokół progu przyjęcia hipotezy, co jest zachowaniem spodziewanym III-C. Bardzo duża częstotliwość nawiązywania połączeń zakończonych porażką powoduje szybki wzrost wartości Ls, a co za tym idzie - bardzo szybką diagnozę hosta jako skanera.

Tabela I zawiera statystyki szybkości (w znaczeniu ilości otrzymanych pakietów) uznania hosta za atakującego, w zależności od stosunku połączeń zakończonych porażką do wszystkich połączeń. Zawiera odpowiednio minimalną, maksymalną oraz średnią liczbę pakietów jaka potrzebna była by zidentyfikować hosta jako atakującego. Im większy jest stosunek połączeń zakończonych porażką, tym szybciej skaner jest wykrywany. Stosunek mniejszy niż 0.2, dla wybranych parametrów P_d oraz P_f uniemożliwia oznakowanie hosta jako atakującego, natomiast przy stosunku większym niż 0.7, odpowiednia diagnoza stawiona jest zawsze, przy średnio małej liczbie pakietów. Warto zauważyć, iż według autorów algorytmu TRW [12] stosunek 0.8 jest wartością, która pozwala na rozróżnienie większości skanerów od pozostałych hostów po 4 - 5 próbach połączeń. Przy takiej wartości stosunku, w eksperymencie uzyskano zbliżoną średnią wartość szybkości wykrywania (po 5.8 pakietach), oraz niską maksymalną liczbę (8 pakietów).

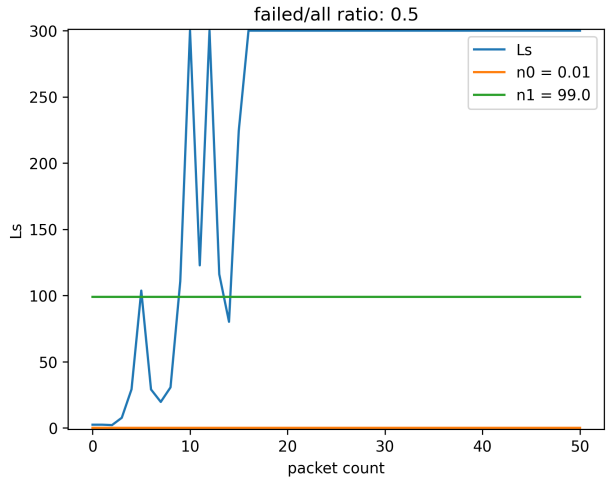
Dla niektórych stosunków, system nie jest wstanie oznaczyć skanera - maksymalna ilość wysłanych pakietów potrzebna do diagnozy, przekracza 50. Wynika to z ograniczeń zaimplementowanego rozwiązania III-C. W zależności od stosunku pakietów, zachowuje się on w następujący sposób:

- próg 0.2 (Rys. 6): dla wszystkich prób, wartość Ls bardzo szybko zanika,

- próg 0.4 (Rys. 7): wartość początkowo oscyluje wokół progu wyboru hipotezy, jednak zanika wraz z kolejnymi pakietami,
- próg 0.5 (Rys. 8): przez cały okres trwania eksperymentu wartość Ls oscyluje między progami wyboru hipotezy, co ilustruje opisaną w III-C własność algorytmu TRW
- próg 0.8 (Rys. 9): wartość bardzo szybko przekracza próg wyboru hipotezy.



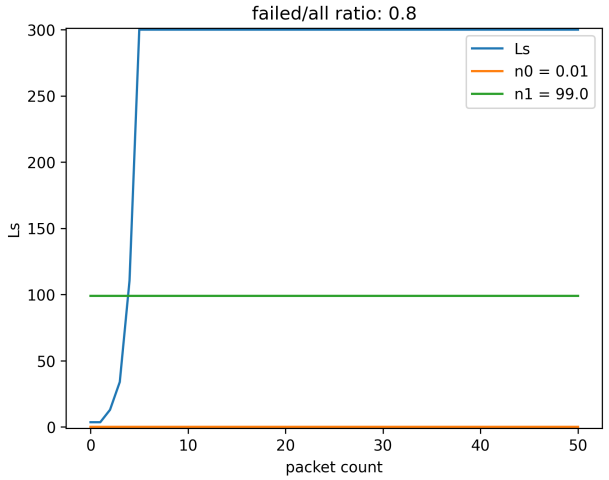
Rysunek 3. Średnia wartość ilorazu prawdopodobieństwa Ls przypisanemu atakującemu hostowi w miare nadchodzenia kolejnych pakietów. Stosunek połączeń 0.2.



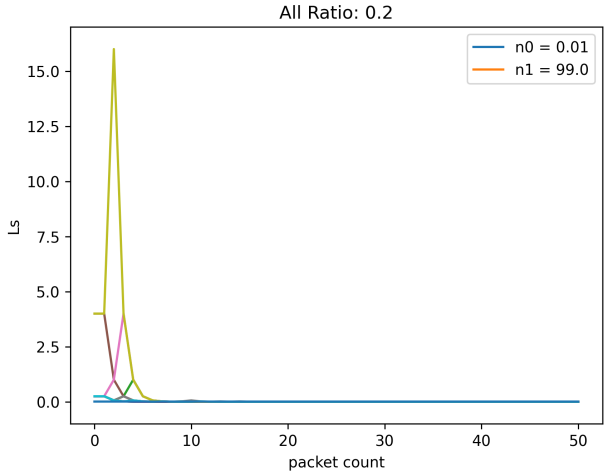
Rysunek 4. Średnia wartość ilorazu prawdopodobieństwa Ls przypisanemu atakującemu hostowi w miare nadchodzenia kolejnych pakietów. Stosunek połączeń 0.5.

VII. Wydajność systemu

Podczas wszystkich przeprowadzonych eksperymentów zużycie procesora na hostcie **H1** powiązanego z procesem badanego systemu oscylowało w okolicach 7%. W celu wyznaczenia granicy wydajności odtworzono scenariusz pesymistyczny pod kątem złożoności obliczeniowej. Do analizy użyto



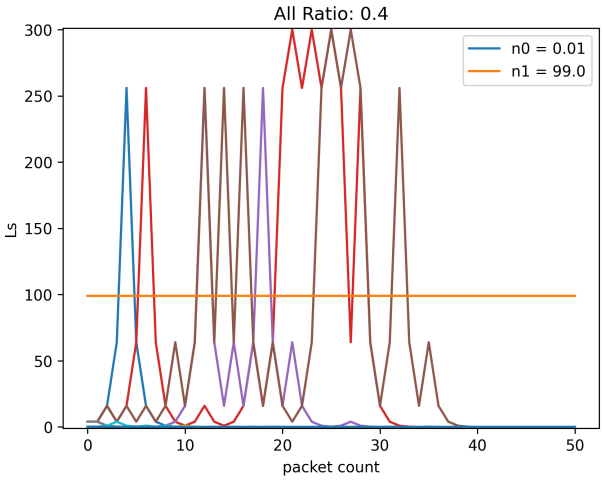
Rysunek 5. Średnia wartość ilorazu prawdopodobieństwa Ls przypisanemu atakującemu hostowi w miare nadchodzenia kolejnych pakietów. Stosunek połączeń 082.



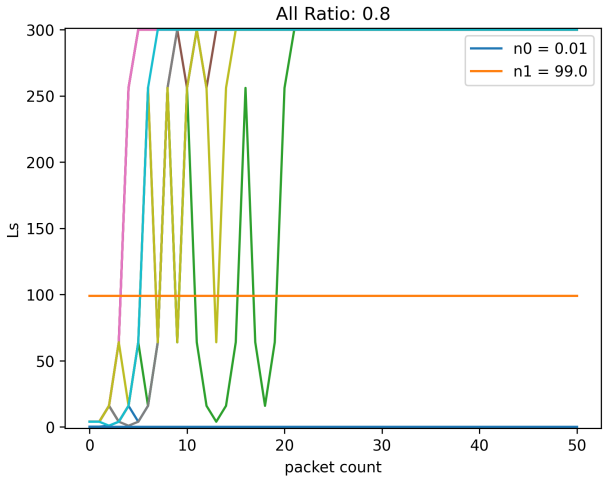
Rysunek 6. Wartości ilorazu prawdopodobieństwa Ls przypisanemu atakującemu hostowi w miare nadchodzenia kolejnych pakietów. Stosunek połączeń 0.2.

sztucznie wygenerowanego ruchu składającego się jedynie z pakietów SYN, które podlegają pełnej analizie przez moduły TRW i TRWP. Próba składała się z 100 tysięcy pakietów bez przerw w wysyłaniu między nimi. Każdy pakiet posiadał inny unikalny adres IP i wysyłany był na jeden port. Dzięki temu każdy wysyłanie wymagało od systemu przeszukanie słownika z obecnymi wpisami, po czym utworzeniu nowego wpisu. Czas przetwarzania wyniósł 30 sekund co daje wynik prędkości przetwarzania 0.3 milisekund na pakiet. W trakcie wykonywania próby zużycie procesora oscylowało w granicach 7-8%, a zużycie pamięci RAM osiągnęło 351MB. Stan sieci zapisywany, co ustaloną w konfiguracji liczbę przetworzeń, wygenerował pliki o rozmiarze 232KB i 5406KB odpowiednio dla modułu TRW oraz TRWP.

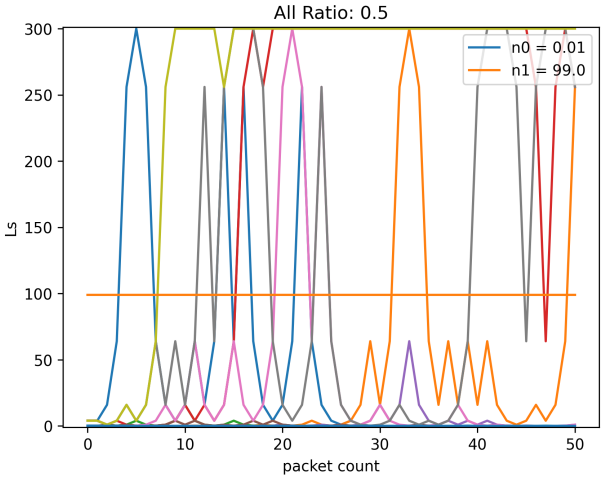
Następnie odtworzono nieco łagodniejszy, jednak bardziej rzeczywisty scenariusz. Ten sam proces, co w przypadku



Rysunek 7. Wartości ilorazu prawdopodobieństwa Ls przypisanemu atakującemu hostowi w miare nadchodzenia kolejnych pakietów. Stosunek połączeń 0.4.



Rysunek 9. Wartości ilorazu prawdopodobieństwa Ls przypisanemu atakującemu hostowi w miare nadchodzenia kolejnych pakietów. Stosunek połączeń 082.



Rysunek 8. Wartości ilorazu prawdopodobieństwa Ls przypisanemu atakującemu hostowi w miare nadchodzenia kolejnych pakietów. Stosunek połączeń 0.5.

pierwszego badania powtórzono dwa razy, gdzie przy drugiej próbie stan systemu został zachowany, a adresacja była taka sama jak w pierwszym przejściu. W ten sposób system musi zarówno dodawać nowe jak i aktualizować już istniejące wpisy. Czas przetwarzania wyniósł 54 sekund co daje wynik prędkości przetwarzania 0.27 milisekund na pakiet. W trakcie wykonywania próby zużycie procesora oscylowało również w granicach 7-8%, a zużycie pamięci RAM osiągnęło 523MB. Stan sieci zapisywany, co ustaloną w konfiguracji liczbę przetworzeń, wygenerował pliki o rozmiarze 232KB i 5406KB odpowiednio dla modułu TRW oraz TRWP.

VIII. Podsumowanie

W niniejszej pracy dokonano opisu zrealizowanej implementacji rozwiązania opartego na algorytmie TRW, pozwalającego na detekcje ataków horyzontalnych skanów portów w ramach protokołu TCP oraz zaproponowano jego rozszerzenie o detekcję skanów wertykalnych. System przetestowano na popularnym zbiorze CICIDS2017, dla którego poprawnie wykryto atak. Przeprowadzono kilka różnych scenariuszy testowych za pomocą powszechnie używanego oprogramowania skanującego jak i autorskich skryptów. System okazał się skuteczny do wykrywania ataków różnego typu (próba pełnego połączenia TCP, atak *ang. stealth*), jednak nieskuteczny w przypadku ataku typu ACK, co było spodziewane. Zbadano reakcje systemu na różne stopnie nasilenia połączeń zakończonych porażką. W przypadku stosunków połączeń zakończonych porażką do wszystkich połączeń o dużych wartościach, system szybko i skutecznie oznaczał skaner. Małe wartości nie pozwalały na wykrycie ataku. Wartości bliskie wartości 0.5 powodowały częste zmiany decyzji sytemu. Wskazuje to na dużą wrażliwość systemu na kolejność i stosunek połączeń zakończonych sukcesem i porażką. Niestety konfrontacja uzyskanych wyników z wynikami autorów artykułu [10] nie była możliwa ze względu na brak dostępnych wykorzystanych zbiorów danych. Warto zauważyć, iż metoda pomimo, że została zaproponowana w 2004 roku wciąż pozwala na detekcję wielu typów skanowań.

Literatura

[1] Ciciflowmeter. https://www.unb.ca/cic/research/applications.html#CICFlowMeter. Dostęp: 2023-06-08.
[2] Nmap. https://nmap.org/. Dostęp: 2023-06-08.
[3] Pyhon. https://www.python.org/. Dostęp: 2023-06-08.
[4] Repozytorium projektu spzc. https://github.com/Ja-Kuba/spzc_projekt. Dostęp: 2023-06-08.
[5] Scapy. https://scapy.net/. Dostęp: 2023-06-08.
[6] Transmission control protocol (rfc 9293). https://www.rfc-editor.org/info/rfc9293. Dostęp: 2023-06-08.
[7] Wireshark. https://www.wireshark.org/. Dostęp: 2023-06-08.
[8] Monowar Bhuyan, Dhruba K Bhattacharyya, and Jugal Kalita. Surveying port scans and their detection methodologies. *The Computer Journal*, 54:1565–1581, 10 2011.
[9] The Canadian Institute for Cybersecurity. Cic-ids2017 dataset. https://www.unb.ca/cic/datasets/ids-2017.html. Dostęp: 2023-06-08.

[10] Carrie Gates, Joshua J McNutt, Joseph B Kadane, and Marc I Kellner. Scan detection on very large networks using logistic regression modeling. In *11th IEEE Symposium on Computers and Communications (ISCC'06)*, pages 402–408. IEEE, 2006.

[11] Richard Harang and Peter Mell. Evasion-resistant network scan detection. *Security Informatics*, 4, 12 2015.

[12] Jaeyeon Jung, Vern Paxson, Arthur W Berger, and Hari Balakrishnan. Fast portscan detection using sequential hypothesis testing. In *IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004*, pages 211–225. IEEE, 2004.

[13] Peter Mell and Richard Harang. Limitations to threshold random walk scan detection and mitigating enhancements. In *2013 IEEE Conference on Communications and Network Security (CNS)*, pages 332–340. IEEE, 2013.

[14] Peter Mell and Richard Harang. Limitations to threshold random walk scan detection and mitigating enhancements. In *2013 IEEE Conference on Communications and Network Security (CNS)*, pages 332–340, 2013.

[15] D.K. Bhattacharyya Monowar H. Bhuyan and J.K. Kalita. Surveying port scans and their detection methodologies. 05 2010.

[16] Arnaud Rosay, Eloïse Cheval, Florent Carlier, and Leroux Pascal. Network intrusion detection: A comprehensive analysis of cic-ids2017. 02 2022.