Jakub Skłodowski*
Piotr Arabas**

Wykorzystanie drzew sufiksowych do efektywnej prezentacji podobieństw sesji z systemu pułapek honeypot

Streszczenie

W artykule przestawiono prototyp systemu do analizy danych z sieci interaktywnych pułapek, tzw. honeypotów. Szczególną uwagę zwrócono na algorytm wyszukiwania podobieństw w zbieranych zapisach sesji ssh. Algorytm ten wyszukuje w sesjach uogólnione wzorce z wykorzystaniem drzew sufiksowych. Wzorce te dzięki zaproponowanej metodzie redukcji mogą być następnie wykorzystane do wygodnej prezentacji zarejestrowanych sesji i efektywnego wyszukiwania. Podsumowanie pracy stanowią przykłady wykorzystania algorytmu.

Słowa kluczowe: honeypoty, malware, analiza sesji, drzewa sufiksowe

* Jakub Skłodowski, inżynier oprogramowania, Dział Systemów Bezpieczeństwa Sieci, Naukowa i Akademicka Sieć Komputerowa – Państwowy Instytut Badawczy, e-mail: Jakub.Sklodowski@nask.pl.
** Piotr Arabas, adiunkt, Dział Systemów Bezpieczeństwa Sieci, Naukowa i Akademicka Sieć Komputerowa – Państwowy Instytut Badawczy, e-mail: Piotr.Arabas@nask.pl.

Wstęp

W ostatnich latach obserwuje się znaczący wzrost liczby ataków cybernetycznych. Naturalną odpowiedzią na taką sytuację jest rozwój systemów zabezpieczeń. Oprócz systemów IDS[1] oraz IPS[2] znaczenie mają również metody zwiększające świadomość sytuacyjną. Umożliwiają to m.in. systemy interaktywnych pułapek sieciowych, tzw. honeypoty. Ich działanie polega na wystawieniu na atak usług przypominających te działające produkcyjnie. Zawierają one typowe podatności, są jednak zabezpieczone przed niepożądanym wykorzystaniem i wyposażone w funkcjonalności umożliwiające rejestrowanie i raportowanie połączeń.

Istnieją dwa główne warianty pułapek sieciowych. W pierwszym są one dostępne z internetu. Należy wtedy liczyć się z dużą liczbą rejestrowanych połączeń. Zgromadzone dane pozwalają poznać metody ataku i istotne podatności. Może to być podstawą zarówno do zmiany konfiguracji urządzeń, jak i automatycznego generowania reguł filtracji dla zapór ogniowych czy sygnatur dla systemów IPS/IDS. W przypadku umieszczenia honeypota w sieci instytucji ataki powinny zdarzać się rzadko i pochodzić z wewnątrz. Zastosowanie to jest bliskie systemowi IDS, który, co jest najważniejszą różnicą, nie analizuje ruchu produkcyjnego, lecz sprowokowane połączenia. Objętość tak gromadzonych danych będzie niewielka, ale ich waga jest zdecydowanie istotniejsza. Mogą one wskazywać na nieprawidłowości w infrastrukturze, a także w ogólnej polityce bezpieczeństwa (np. dopuszczaniu osób nieuprawnionych do wrażliwych systemów).

W obu przypadkach korzyść zależy od użytych metod analizy danych, przy czym ważną rolę odgrywa automatyczna agregacji i wydajna, atrakcyjna prezentacja wyników. Żeby uzmysłowić skalę zagadnienia, wystarczy wspomnieć, że opisywany system był wyposażony w trzy pułapki sieciowe i rejestrował średnio około kilka tysięcy zdarzeń dziennie. Z tego powodu w niniejszym artykule oprócz ogólnej prezentacji prototypu systemu skupiono się na algorytmie wyszukującym podobnych sekwencji poleceń w zarejestrowanych sesjach ssh. Algorytm ten należy traktować przede wszystkim jako metodę agregacji sesji pozwalającą na ich wydajne przeszukiwanie. Należy podkreślić,

1 IDS – Intrusion Detection System – system wykrywający atak na systemy teleinformatyczne.

2 IPS – Intrusion Prevention System – system zapobiegający atakowi na systemy teleinformatyczne.

że dla honeypotów waga specyficznych fragmentów skryptu, np. adresów IP, jest mniejsza niż w przypadku IDS. W omawianym przypadku istotniejsze jest wskazanie ogólnych cech ataku, co może prowadzić do określenia jego sygnatury. Nie oznacza to oczywiście rezygnacji z obserwowania tych szczegółów. W opisywanym systemie zaproponowano w tym celu inne analizy, np. sieć powiązań między adresami IP.

Definicja zadania i sposobu jego rozwiązania

Rozważanym w artykule zagadnieniem jest wyszukiwanie podobieństw w sesjach ssh zarejestrowanych przez pułapkę sieciową cowrie[3]. Jak zauważono w trakcie eksperymentów, sesje często różnią się nieznacznymi fragmentami, np. adresami IP użytymi jako parametry poleceń systemu. Potwierdzają to także inne prace[4]. Oznacza to, że są one zazwyczaj wynikiem użycia tej samej metody ataku. Stąd propozycja narzędzia umożliwiającego wygodne wyszukiwanie sesji, dokonującego generalizacji poprzez pomijanie argumentów i wstępną klasyfikację poleceń w zależności od ich funkcji. Oryginalnym wkładem jest wykorzystanie drzew sufiksowych do wyszukiwania wspólnych podciągów w analizowanym zbiorze sesji, a także autorski algorytm redukcji zawierających się podciągów i wykorzystanie ich do wyszukiwania i prezentacji sesji. Zaletą jest brak fazy uczenia, a co za tym idzie, potrzeby przygotowania zbioru uczącego. Pozwala to na grupowanie nowych, nieobserwowanych wcześniej ataków.

Analiza danych z honeypotów

Pułapki sieciowe wymagają analizowania tysięcy zdarzeń dziennie[5]. Co istotne, można spodziewać się, że wiele z nich będzie zawierało powtarzające się informacje takie, jak: źródłowe adresy IP, użyte loginy i hasła, charakterystyczne

3 https://github.com/cowrie/cowrie [dostęp: 7.01.2023].

4 J.M. Jorquera Valero, M. Pérez, A. Huertas, G. Martinez Perez, *Identification and classification of cyber threats through SSH honeypot systems* [w:] B.B. Gupta, S. Srinivasagopalan, *Handbook of Research on Intrusion Detection Systems*, Hershey, PA 2020; O. Navarro Ferrer, *Analysis of reinforcement learning techniques applied to honeypot systems*, Master's thesis, Universitat Oberta de Catalunya, Barcelona 2021.

5 P. Rabadia, C. Valli, A. Ibrahim, Z. Baig, *Analysis of attempted intrusions: intelligence gathered from ssh honeypots* [w:] *The 15th Australian Digital Forensics Conference*, Perth 2017.

komendy czy pobierane pliki[6]. Przykładami analizatorów dla pułapki sieciowej cowrie mogą być dostępne na licencji open-source pakiety analyze-cowrie-log[7] czy cowrie-log-analyzer[8] gromadzące hasła i wyznaczające podstawowe statystyki, a także realizujące geolokalizację, wyszukiwanie pojedynczych komend czy analizę czasu trwania połączeń. Stosunkowo rozbudowany system t-pot[9] zawiera atrakcyjny wizualnie interfejs wykorzystujący pakiet Kibana[10]. Jego istotną funkcjonalnością jest przeszukiwanie danych z pomocą Elasticsearch[11]. Przykładowe analizy to geolokalizacja i rozpoznawanie systemu operacyjnego, z którego przeprowadzono atak[12], lub wyznaczanie statystyki popularności loginów i haseł[13].

Nieco bardziej zaawansowanym zastosowaniem może być generowanie sygnatur ataku typu brute-force na podstawie częstotliwości połączeń[14]. Jednakże statystyki nie mówią wiele o naturze zdarzeń czy poziomie zagrożenia. Pomagają tutaj metody klasyfikacji, wśród których można wyróżnić dwie najważniejsze grupy. Pierwsza wykorzystuje cechy połączenia raportowane przez pułapkę podobnie jak opisane wyżej statystyki. Druga skupia się na analizie sesji. Za warianty pierwszego podejścia można uznać wykorzystanie do klasyfikacji sesji maszyny wektorów podpierających (SVM)[15], a także system[16], w którym zdefiniowano 52 cechy, w tym oprócz adresu źródła czy lokalizacji geograficznej występowanie wybranych komend w sesji ssh. Należy zauważyć, że

6 K. Lasota, E. Niewiadomska-Szynkiewicz, A. Kozakiewicz, *Adaptacja rozwiązań honeypot dla sieci czujników*, „Studia Informatica" 2012, t. 33, nr 1, s. 139–148.
7 https://github.com/vwefnab/analyze-cowrie-log [dostęp: 7.01.2023].
8 https://github.com/jasonmpittman/cowrie-log-analyzer [dostęp: 7.01.2023].
9 https://github.com/telekom-security/tpotce [dostęp: 7.01.2023].
10 https://github.com/elastic/kibana [dostęp: 7.01.2023].
11 https://github.com/elastic/elasticsearch [dostęp: 7.01.2023].
12 N. Memari, S. Hashim, K. Samsudin, *Network probe patterns against a honeynet in Malaysia*, „Defence S and T Technical Bulletin" 2015, t. 8, nr 1, s. 63–75.
13 M. Boddy, *Exposed: Cyberattacks on cloud honeypots*, 2019, https://assets.sophos.com/X24WTUEQ/at/rgbjvgnx6qwwj7wvx764rmbn/sophos-exposed-cyberattacks-on-cloud-honeypots-wp.pdf [dostęp: 7.01.2023]; C. Kelly, N. Pitropakis, A. Mylonas, S. McKeown, W.J. Buchanan, *A comparative analysis of honeypots on different cloud platforms*, „Sensors" 2021, t. 21, nr 7.
14 E. Satria, T.P.S. Huda, M. Iqbal, F. Sarjana, *The investigation on cowrie honeypot logs in establishing rule signature snort* [w:] *International Conference on Agricultural Technology, Engineering, and Environmental Sciences (ICATES)*, Banda Aceh 2020.
15 J. Martinez, M. Pérez, A. Ruiz-Martínez, *A novel machine learning-based approach for the detection of ssh botnet infection*, „Future Generation Computer Systems" 2021, t. 115, s. 387–396.
16 B. Wang, J. Chen, C. Yu, *An ai-powered network threat detection system*, „IEEE Access" 2022, t. 10, s. 1–1.

wykorzystanie komend jako jednej z cech sesji zbliża tę metodę do drugiej grupy bazującej zazwyczaj na algorytmach analizy języka naturalnego. Również metoda opisana przez Febriana Setianto i współpracowników[17] jest hybrydą – używa ona algorytmu GPT2 do analizy dzienników honeypota cowrie, co jest próbą wydobycia cech przez rozpoznanie składni. System CYBEX-P[18] oprócz typowych analiz próbuje prognozować kolejne polecenie w połączeniu. Co ciekawe, prosty model wykorzystujący odległość Levenshteina daje w badanym przypadku lepsze rezultaty niż złożona sieć LSTM. Innym przykładem jest zastosowanie n-gramów do wyszukiwania podobnych sesji[19]. Reprezentacja sesji w postaci grafu[20] pozwala odwzorować powiązania komend, przy czym generalizację uzyskano dzięki pomijaniu argumentów. Kolejnym uogólnieniem jest wprowadzenie klas poleceń odpowiedzialnych za wybrane działania, np. modyfikację konfiguracji systemu czy wstępne ustalanie jego przydatności do dalszego wykorzystania[21]. Tak zdefiniowane klasy stanowią cechy wykorzystywane do klasyfikacji.

Drzewa sufiksowe

Zadanie znajdowania podobieństwa sesji można sprowadzić do wyszukiwania powtarzających się wzorców w postaci podciągów złożonych z określonej liczby znaków. W tym celu często buduje się struktury grafowe, tzw. drzewa sufiksowe (przypadek pojedynczego słowa) oraz uogólnione drzewa sufiksowe (dla zbioru słów)[22]. Metoda Ukkonena umożliwia zastosowanie wielu algorytmów, w tym znalezienie najdłuższego wspólnego podciągu czy zliczenie wszystkich wystąpień danej sekwencji.

Sposób tworzenia drzew sufiksowych przedstawia przykład z jednym słowem – „bazar" (patrz rys. 1). Budowa drzewa ma charakter inkrementalny, co

17 F. Setianto i in., *Gpt-2c: A gpt-2 parser for cowrie honeypot logs*, 2021, https://arxiv.org/abs/2109.06595 [dostęp: 7.01.2023].

18 F. Sadique, S. Sengupta, *Analysis of attacker behavior in compromised hosts during command and control* [w:] *ICC 2021 – IEEE International Conference on Communications*, Montreal 2021, s. 1–7.

19 P. Dumont, R. Meier, D. Gugelmann, V. Lenders, *Detection of malicious remote shell sessions* [w:] *2019 11th International Conference on Cyber Conflict (CyCon)*, t. 900, Tallinn 2019, s. 1–20.

20 O. Navarro Ferrer, op. cit.

21 J.M. Jorquera Valero, M. Pérez, A. Huertas, G. Martinez Perez, op. cit.

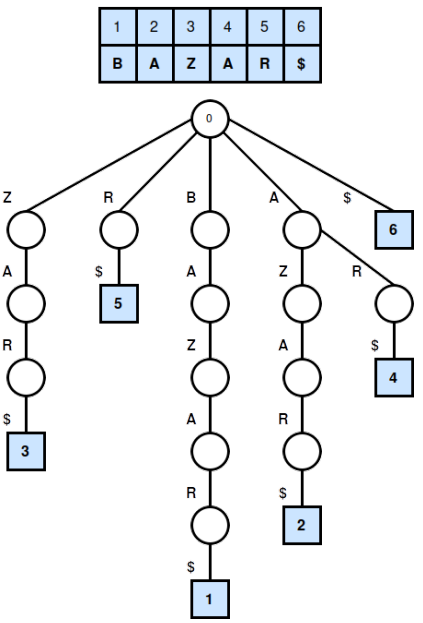
22 E. Ukkonen, *On-line construction of suffix trees*, „Algorithmica" 1995, t. 14, nr 3, s. 249–260.

pozwala na zachowanie liniowej złożoności obliczeń. W pierwszym kroku do słowa należy dodać znak niewystępujący w alfabecie np. *$* symbolizujący koniec – w podanym przykładzie *bazar$*. Punktem startowym grafu jest *korzeń* (ang. root) z indeksem 0. W każdej z n+1 iteracji należy wybrać kolejną literę (w przykładzie *b*, *ba*, *baz*, ...) i rozbudować drzewo z poprzedniej iteracji według reguł:

1) jeżeli i-ta ścieżka od korzenia kończy się *liściem* (ang. leaf node) – i-ty znak jest ostatnim elementem podciągu – to nowy znak i+1 należy dodać na końcu;

2) jeżeli i-ta ścieżka od korzenia jest zakończona *węzłem pośrednim* (istnieje kontynuacja ścieżki) i kolejne znaki są różne od nowego znaku i+1, to należy utworzyć nowe rozgałęzienie i-tego węzła, którego końcem będzie znak i+1;

3) jeżeli i-ta ścieżka od korzenia jest zakończona *węzłem pośrednim*, ale kolejny znak jest taki sam jak nowy znak i+1, to należy przejść do kolejnej iteracji.

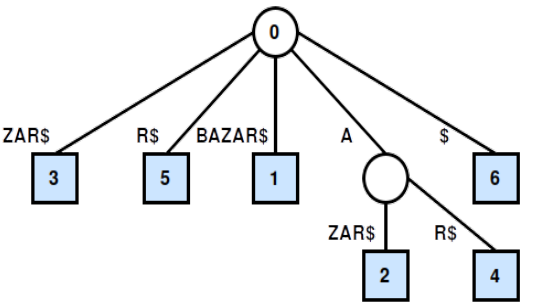


Źródło: Opracowanie własne.

Rys. 1. Konstrukcja drzewa sufiksowego dla słowa „bazar"

Ostatnim etapem jest upraszczanie struktury, rozpoczynając od końcówek. Jeżeli dany *liść* jest jedyną kontynuacją *rodzica* (nie ma więcej rozgałęzień), to węzły te można połączyć, pamiętając o aktualizacji etykiety krawędzi.

Operację należy powtórzyć dla wszystkich końcówek grafu. Końcową postać drzewa sufiksowego przedstawia rysunek 2. Liczby umieszczone w błękitnych kwadratach odpowiadają indeksowi litery, od której rozpoczyna się każda gałąź. Przedstawioną metodę można uogólnić do zadania budowy drzewa sufiksowego dla zestawu słów.



Źródło: Opracowanie własne.

Rys. 2. Końcowa postać drzewa sufiksowego

Struktura systemu

W strukturze prototypowego systemu można wyróżnić trzy warstwy:

1) pułapek sieciowych;
2) zbierania i dystrybucji danych;
3) analizy i prezentacji danych.

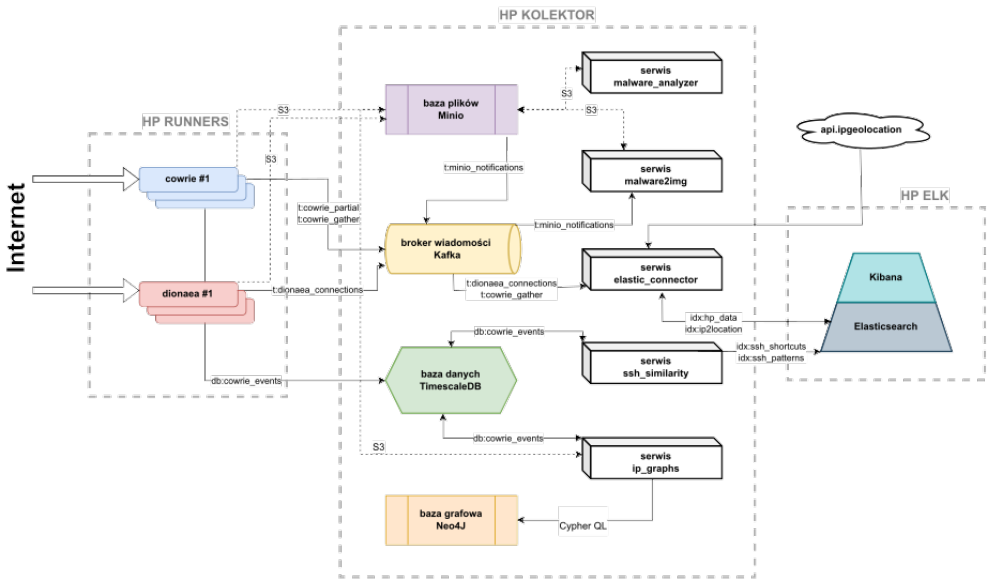
W pierwszej warstwie znajdują się pułapki sieciowe (honeypoty). W obecnej wersji wykorzystywane są dwa typy – cowrie emulujący protokół ssh oraz dionaea[23] odwzorowująca m.in. ftp, SMB, mysql, mssql, MongoDB, memcache. W warstwie zbierania i dystrybucji danych wykorzystano: broker Kafka do zbierania i udostępniania rejestrowanych zdarzeń, bazę plikową MinioDB do przechowywania przechwyconych złośliwych plików (malware), a także wyników wykonanych na nich analiz oraz bazę TimescaleDB[24] dla próbek dotyczących zarejestrowanych sesji. Dodatkowo użyto bazy grafowej Neo4J[25] do przechowywania sieci powiązań między adresami IP i URL oraz Elasticsearch,

23 *Welcome to dionaea's documentation!*, https://dionaea.readthedocs.io/en/latest/ [dostęp: 7.01.2023].
24 https://www.timescale.com/ [dostęp: 7.01.2023].
25 https://neo4j.com/ [dostęp: 7.01.2023].

którego funkcją jest filtrowanie i agregacja danych dotyczących logowań do pułapek, poleceń i skrótów sesji ssh. Analizatory danych obejmują:

1) analizator złośliwych plików: malware_analyzer i malware2img;
2) analizator danych logowania wykonujący ekstrakcję poleceń ssh i geolokalizację elastic_connector;
3) tworzenie sieci powiązań między adresami – ip_graphs;
4) wyszukiwanie podobieństw w sesjach ssh – ssh_similarity.

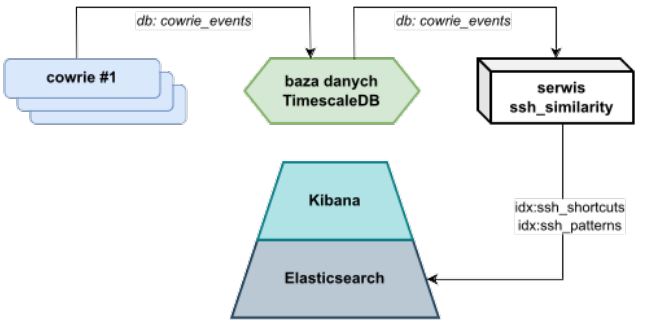
Prezentacja danych wykonana jest z pomocą narzędzia Kibana, które w połączeniu z Elasticsearch umożliwia efektywne przeszukiwanie danych. Schemat systemu przedstawia rysunek 3.



Źródło: Opracowanie własne.

Rys. 3. Schemat prototypu systemu

W dalszej części artykułu skupiono się na zaproponowanym sposobie analizy i prezentacji rejestrowanych przez system sesji ssh. Zbudowanie odpowiedniego narzędzia zmniejszającego nakład pracy operatora systemu umożliwia oddzielenie przypadków typowych, względnie będących ich niewielką modyfikacją, od szczególnie ciekawych i potencjalnie niebezpiecznych wariantów nowych, występujących pojedynczo. Schemat podsystemu przedstawia rysunek 4.



Źródło: Opracowanie własne.

Rys. 4. Schemat przepływu danych usługi ssh_similarit

Honeypot cowrie

Honeypot cowrie emuluje serwer ssh. Pozwala on na stosunkowo swobodną interakcję, np. operacje na plikach czy pobieranie danych z pomocą wget i ftp. Skutki takich operacji są ograniczone tak, żeby zminimalizować możliwość uszkodzenia pułapki bądź wykorzystanie jej do kontynuacji ataku. Zapis sesji jest zarówno przechowywany w pliku dziennika, jak i udostępniany z pomocą specyficznych protokołów, w tym hpfeeds. Ograniczenia tego protokołu (m.in. brak trwałości danych) spowodowały, że zastąpiono go bezpośrednią komunikacją z bazą TimescaleDB przechowującą zapis sesji. Dodatkowym rozszerzeniem jest tzw. plugin dla brokera Kafka[26]. Publikuje on dane zawierające podsumowanie sesji ssh.

TimescaleDB

Baza TimescaleDB[27] zawiera tabelę cowrie_events gromadzącą zdarzenia z honeypotów cowrie w postaci szeregów czasowych. Oprócz elementów wspólnych dla różnych rodzajów zdarzeń (np. stempel czasowy czy identyfikator honeypota) pozostałe dane przechowywane są w strukturze słownikowej zapisywanej w formacie binarnym. Takie podejście pozwala na zawarcie kompletu informacji, z zachowaniem możliwości przeszukiwania kluczy słownika

26 https://kafka.apache.org/ [dostęp: 7.01.2023].
27 https://www.timescale.com/ [dostęp: 7.01.2023].

z wykorzystaniem rozszerzonego języka SQL. Zapis zdarzeń w formie szeregów czasowych pozwala na tworzenie zapytań SQL operujących na wskazanym przedziale czasowym czy wybranie konkretnego typu zdarzenia z 18 generowanych, m.in. logowanie czy wgranie pliku.

Poszukiwanie wzorców sesji ssh

Obserwowane sesje wykazują znaczne podobieństwa. Korzystne byłoby grupowanie albo wyszukiwanie sesji ssh, które łączy wspólny wzorzec. W tym celu zaprojektowano serwis ssh_similarity pobierający zdarzenia z bazy TimescaleDB oraz zwracający wyniki analizy do indeksów Elasticsearch (zob. rys. 4).

Algorytm działania sprowadza się do połączenia raportowanych przez pułapkę zdarzeń w spójny zapis sesji, a następnie redukcji do skrótu literowego. Do rozpoznawania składni interpretera bash wykorzystano analizator shlex[28] wykonujący tokenizację i pozwalający odróżnić polecenia systemu od argumentów. Polecenia są kwalifikowane do 14 klas oznaczonych literami od A do N, wśród zaś argumentów wyróżniono:

1) ścieżki do plików: litera "z";
2) opcje poleceń: litera „x";
3) adresy URL: litera „y";
4) adresy IP: litera „v",
5) pozostałe, nierozpoznane: znak „;".

Tak powstałe skróty literowe są umieszczane w indeksie Elasticsearch. Podejście takie pozwala na uogólnienie zapisu sesji, tzn. utożsamianie sesji różniących się jedynie argumentami poleceń lub używających równoważnych poleceń. Przykładem mogą być „curl" i „wget", służące do pobierania plików. Kontekst nadawany poleceniom dzięki podziałowi na kategorie umożliwia drastyczną redukcję wymiarowości danych – przestrzeń 172 poleceń oraz nieograniczona liczba argumentów są zastępowane przez 19 znaków, przy czym informacja o kolejności poleceń jest zachowywana. Przykład tłumaczenia przedstawiono na rysunku 5. Na potrzeby analizy przygotowano słownik mapujący 172 najpopularniejsze polecenia systemu Linux na 14 kategorii:

28 https://docs.python.org/3/library/shlex.html [dostęp: 7.01.2023].

Tabela 1. Kategorie poleceń usługi ssh_similarity

Kategoria	Symbol	Przykład
Compression	A	gzip
Configuration	B	adduser
Display	C	cat
File operations	D	chmod
Help	E	man
Info	F	who
Install	G	pip
Networking	H	iptables
Programming	I	gcc
Shells	J	sh
System	K	kill
Text processing	L	sed
Transfer	M	wget
Utils	N	bc

Źródło: Opracowanie własne.

```
cd /tmp || cd /run || cd /; wget
http://109.206.241.200/Gummybins.sh; chmod 777
Gummybins.sh; sh Gummybins.sh; tftp 109.206.241.200 -c
get Gummytftp1.sh; chmod 777 Gummytftp1.sh; sh
Gummytftp1.sh; tftp -r Gummytftp2.sh -g 109.206.241.200;
chmod 777 Gummytftp2.sh; sh Gummytftp2.sh; rm -rf
Gummybins.sh Gummytftp1.sh Gummytftp2.sh; rm -rf *
```

```
DzDzDzMyD__J_Mvx__D__J_Mx_xvD__J_Dx___Dx
```

Źródło: Opracowanie własne.

Rys. 5. Przykład tłumaczenia sesji ssh
na skróty literowe – usługa ssh_similarity

Wyszukiwanie powtarzalnych wzorców w przetransformowanych sesjach ssh wykorzystuje koncepcję uogólnionych drzew sufiksowych. Danymi wejściowymi programu zaimplementowanego w języku Scala na bazie biblioteki gstlib[29] jest zestaw skrótów sesji ssh. Skrypt buduje w pierwszym kroku graf w postaci drzewa sufiksowego, a następnie znajduje wszystkie istniejące

29 G. Dubuisson Duplessis i in. *Utterance retrieval based on recurrent surface text patterns* [w:] *European Conference on Information Retrieval*, Aberdeen 2017, s. 199–211.

wspólne podciągi znaków oraz ich liczebność rozumianą jako liczba słów z zestawu zawierająca dany wzorzec. Wygenerowany zestaw jest następnie redukowany, ponieważ część wzorców zawiera się w pozostałych. Ignorowane są także wzorce złożone z pojedynczego znaku. Procedurę zilustrowano na abstrakcyjnym przykładzie zestawu słów „mango, banan, ananas":

1) budowa drzewa sufiksowego dla wybranego zestawu;

2) wygenerowanie listy wszystkich wspólnych podciągów – [(‚A', 3), (‚N', 3), (‚AN', 3), (‚NA', 2), (‚ANA', 2), (‚NAN', 2), (‚ANAN', 2)];

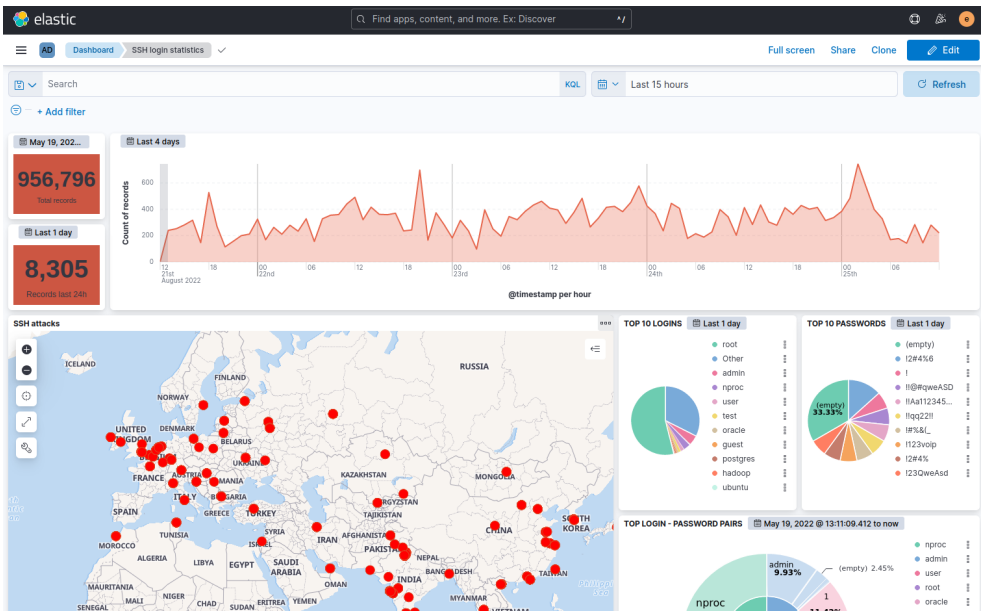
3) eliminacja wzorców o długości jednego znaku – [(‚AN', 3), (‚NA', 2), (‚ANA', 2), (‚NAN', 2), (‚ANAN', 2)];

4) eliminacja wzorców zawartych w innych wzorcach tylko wtedy, kiedy ich liczebność jest równa – [(‚AN', 3), (‚ANAN', 2)].

Odfiltrowane wzorce są zapisywane w Elasticsearch razem z częstotliwością występowania. Dzięki silnikowi wyszukiwarki możliwe jest znalezienie wszystkich sesji ssh pasujących do wykrytych sekwencji.

Elasticsearch i Kibana

Platformę Elasticsearch wykorzystano do indeksowania i wyszukiwania danych. Wbudowane mechanizmy dają wiele możliwości w zakresie przeszukiwania tekstu, filtrowania, agregacji danych i tworzenia statystyk. Elasticsearch pozwala na rozróżnienie kilkunastu typów danych, w tym stempli czasowych, adresów czy różnych formatów zapisu liczb. Szczególnie wykorzystanie formatu text pozwala na efektywne wyszukiwanie przez częściowe dopasowanie frazy, format keyword zaś na zwracanie jedynie wyników pełnego dopasowania oraz tworzenie agregacji i statystyk. Do wizualizacji danych użyto narzędzia Kibana. Jest to aplikacja webowa, która oprócz możliwości wygodnego przeglądania danych udostępnia wiele możliwości administrowania platformą, instalację dodatków, a przede wszystkim tworzenie interaktywnych ekranów (tzw. dashboardów) z wykresami. Przykładowy widok zaprezentowano na rysunku 6.

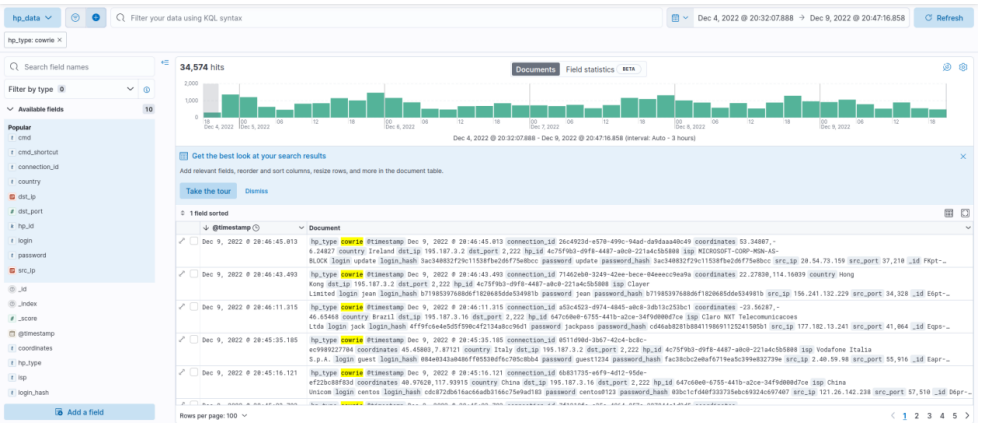


Źródło: Opracowanie własne.

Rys. 6. Przykładowy widok interfejsu graficznego przedstawiający statystyki logowań do honeypotów

Przykładowe wyniki

W celu zademonstrowania możliwości wykorzystania opisanej metody analizy sesji ssh posłużono się danymi zebranymi między 4 a 9 grudnia 2022 roku. Omawiany okres zawiera dni powszednie od poniedziałku do piątku. W tym czasie zarejestrowano 4923 sesje ssh. Na rysunku 7 przedstawiono interfejs graficzny prototypu. Widoczne są wszystkie zdarzenia zarejestrowane w omawianym okresie, w tym m.in. nieudane próby logowania, dlatego ich liczba (ponad 34 tys.) jest znacznie większa niż liczba sesji. Można zauważyć, że natężenie zdarzeń jest stosunkowo duże, rzędu kilkuset na godzinę, przy czym w godzinach nocnych daje się zaobserwować niewielki wzrost aktywności.

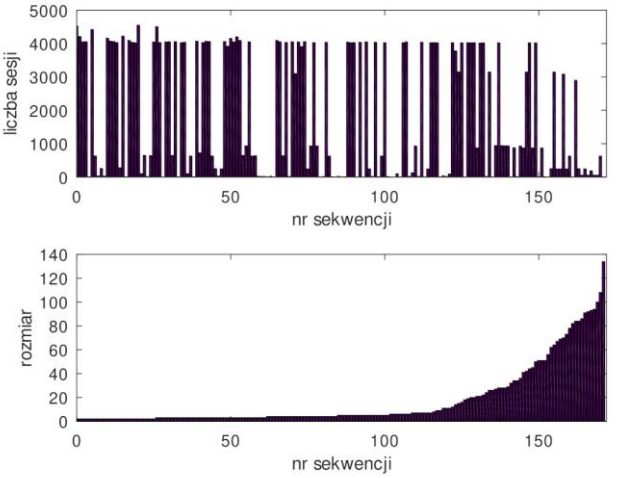


Źródło: Opracowanie własne.

Rys. 7. Widok interfejsu graficznego w trybie przeglądania zdarzeń, w górnym oknie wykres częstotliwości zdarzeń w badanym przedziale czasu

Po przetworzeniu 4923 zapisanych sesji uzyskano 172 sekwencje skrótów. Liczba ta jest stosunkowo mała, co świadczy o wydajnym działaniu procedury redukcji, może również być wynikiem małego zróżnicowania ataków, co było obserwowane podczas przeglądania danych. Dane przedstawione na rysunku 7, mimo że nie dotyczą bezpośrednio sesji, potwierdzają powtarzanie przez atakujących tych samych procedur. Widoczne są tam uporczywe próby ataku słownikowego, co ciekawe, wykonywane z wielu adresów źródłowych. W wyniku zastosowania redukcji skrótów uzyskano dość znamienny rozkład ich długości – znaczną liczbę sekwencji krótkich, które gwałtownie przechodzą w sekwencje dość długie, rzędu 100 lub więcej symboli (zob. rys. 8).

Zgodnie z oczekiwaniem wiele sekwencji krótkich odznacza się bardzo dużą liczbą dopasowań do sesji, przekraczającą często tysiąc. W większości przypadków oznacza to, że zawierają mało znaczące, powszechnie używane (często wielokrotnie w sesji) ciągi poleceń. Przykładem może być występująca w zbiorze 4038 razy sekwencja „Cz_L__Lx", która może odpowiadać poleceniom „cat /proc/cpuinfo | grep name | wc -l", ale również „cat /proc/cpuinfo | grep name | head -n}". Widoczna tu jest istotna właściwość zaproponowanej metody – kodowanie z pomocą słownika złożonego z kilkunastu symboli nie jest jednoznaczne, jednej sekwencji symboli może odpowiadać wiele sekwencji poleceń. Jak widać, podejście takie pozwala na uogólnianie znaczenia zapisu sesji – przytoczone powyżej przykłady mają podobny sens, sprowadzają się do poszukiwania informacji w pewnym pliku – wskazuje to przede wszystkim sekwencja „Cz".



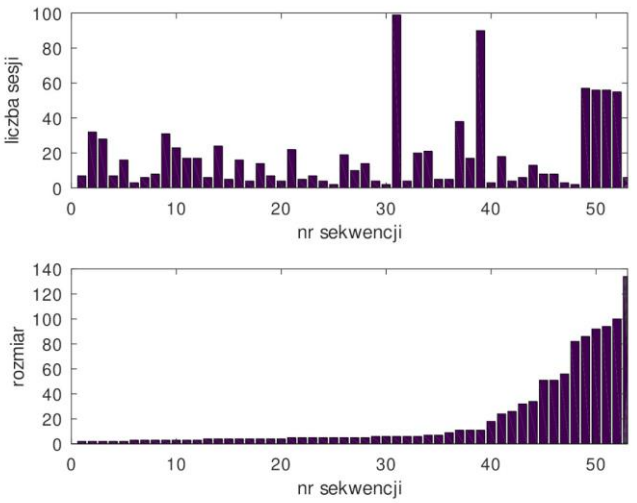
Źródło: Opracowanie własne.

Rys. 8. Liczba dopasowań sesji do kolejnych sekwencji skrótów (wykres górny) oraz długość sekwencji (wykres dolny)

O wiele bardziej symptomatyczne jest wielokrotne występowanie sekwencji długich, np. sekwencja „Cz_L__LxC__B_Jcz_L__Cx__I_Fx_L__I_DxKDKDBxF-FxCz_L__L__LxKFFxF_L_D_Dx__D__C___Dx___D" o numerze porządkowym 162 występuje w zbiorze sesji 2897 razy. Oznacza to, że analizowane sesje są rzeczywiście bardzo podobne do siebie. Przykładowe sesje odpowiadające podanemu skrótowi mogą zawierać następujące polecenia: „cat /proc/cpuinfo | grep name | wc -l echo „root:weQcvQXFk4Rn" | chpasswd"[30] lub „cat /proc/cpuinfo | grep name | wc -l echo „root:gspt2HdIhFWy" | chpasswd"[31]. Jak widać, znowu udało się dokonać generalizacji – obie przytoczone sesje zawierają próbę zmiany hasła administratora systemu. Zostało to wychwycone, mimo że podane hasła są całkowicie różne.

Żeby przyjrzeć się sekwencjom występującym rzadziej, wynikowy zbiór przefiltrowano, ograniczając się do tych, które uzyskały nie więcej niż 100 dopasowań (patrz rys. 9).

30 Zapis skrócono, aby nie zaciemniać tekstu.
31 Tak jak w poprzednim przypadku podano tylko początkowy fragment, ciąg dalszy jest identyczny w obu przypadkach.



Źródło: Opracowanie własne.

Rys. 9. Mniej popularne sekwencje – wykres górny przedstawia liczbę dopasowań sesji, wykres dolny – długość sekwencji. Ograniczono się do sekwencji z nie więcej niż 100 dopasowaniami

Na pierwszy plan wybijają się dwie sekwencje o największej liczbie dopasowań. Pierwsza z nich – „Fxxxxx", pasująca do 99 sesji, nie wydaje się szczególnie ciekawa, odpowiada wywołaniu pojedynczej komendy „uname -s -v -n -r -m". Polecenie to samo w sobie nie jest szczególnie groźne, służy bowiem do sprawdzenia wersji systemu operacyjnego oraz architektury komputera, może jednak być użyte do wstępnego rozpoznania celu, w który ma być wymierzony atak. Druga sekwencja jest nieco dłuższa: „FxB__B__Cz" i odpowiada zapisowi sesji: „uname -a; sudo hive-passwd set 234tg3ji24hj34hju345huj; sudo hive-passwd ij24ghji34hij53ji45h; cat /hive-config/rig.conf". Występuje ona w badanym zbiorze 90 razy, co jest o tyle ciekawe, że zawiera prawdopodobnie próbę przejęcia kontroli nad specyficznym systemem klastrowym przeznaczonym do obliczeń z dziedziny sztucznej inteligencji[32].

Jeżeli chodzi o sekwencje występujące najrzadziej, to można wskazać np. następującą: „C_D___". Mimo niewielkiej długości została ona zarejestrowana tylko dwa razy i odpowiada instrukcjom „cat > au; chmod +x au; ./au &". Podany przykład ilustruje zdolność algorytmu do wykrywania sesji zawierających

32 https://hiveon.com/os/ [dostęp: 7.01.2023].

nietypowe operacje. W istocie, wydaje się, że właśnie rzadko występujące skróty powinny być sprawdzane w pierwszej kolejności, potencjalnie bowiem zawierają nieobserwowane wcześniej i potencjalnie niebezpieczne połączenia.

Wnioski

W artykule został przedstawiony prototyp systemu zbierającego i analizującego dane z pułapek sieciowych, a w szczególności jego część zajmująca się analizą i prezentacją sesji ssh zarejestrowanych przez honeypot cowrie. Zaprezentowane wyniki wskazują na użyteczność zaproponowanej metody bazującej na kodowaniu poleceń systemu operacyjnego w postaci symboli literowych i wyszukiwaniu wspólnych podciągów z użyciem drzew sufiksowych. Zastosowanie zaproponowanego algorytmu redukcji pozwala zmniejszyć liczbę podciągów do związanych ze znacząco różnymi sesjami. Należy podkreślić, że wynikowa liczba uzyskanych w ten sposób skrótów jest znacznie mniejsza niż liczba zarejestrowanych sesji. Pozwala to traktować tak uzyskany wynik jako rodzaj grupowania przeprowadzanego bez nadzoru. Co ważne, metoda ta nie wymaga zbioru uczącego oraz wstępnego określenia liczby poszukiwanych klas. Pozwala to znajdować sesje występujące rzadko czy wręcz rejestrowane pierwszy raz od uruchomienia systemu. Podczas dalszego ulepszania prototypu przewidywane jest poprawienie wygody wyszukiwania wzorców poprzez implementację graficznego narzędzia zastępującego dotychczasowy interfejs wyszukiwania Elasticsearch, a także silniejsze powiązanie z wynikami dostarczanymi przez pozostałe analizatory.

Bibliografia

Boddy M., *Exposed: Cyberattacks on cloud honeypots*, 2019, https://assets.sophos.com /X24WTUEQ/at/rgbjvgnx6qwwj7wvx764rmbn/sophos-exposed-cyberattacks-on-cloud-honeypots-wp.pdf [dostęp: 7.01.2023].

Dubuisson Duplessis G. i in., *Utterance retrieval based on recurrent surface text patterns* [w:] *European Conference on Information Retrieval*, Aberdeen 2017.

Dumont P., Meier R., Gugelmann D., Lenders V., *Detection of malicious remote shell sessions* [w:] *2019 11th International Conference on Cyber Conflict*, t. 900, Tallinn 2019.

Jorquera Valero J.M., Pérez M., Huertas A., Martinez Perez G., *Identification and classification of cyber threats through SSH honeypot systems* [w:] Gupta B.B., Srinivasagopalan S., *Handbook of Research on Intrusion Detection Systems*, Hershey, PA 2020.

Kelly C., Pitropakis N., Mylonas A., McKeown S., Buchanan W.J., *A comparative analysis of honeypots on different cloud platforms*, „Sensors" 2021, t. 21, nr 7.

Lasota K., Niewiadomska-Szynkiewicz E., Kozakiewicz A., *Adaptacja rozwiązań honeypot dla sieci czujników*, „Studia Informatica" 2012, t. 33, nr 1.

Martinez J., Pérez M., Ruiz-Martínez A., *A novel machine learning-based approach for the detection of ssh botnet infection*, „Future Generation Computer Systems" 2021, t. 115.
Memari N., Hashim S., Samsudin K., *Network probe patterns against a honeynet in Malaysia*, „Defence S and T Technical Bulletin" 2015, t. 8, nr 1.
Navarro Ferrer O., *Analysis of reinforcement learning techniquesapplied to honeypot systems*," Master's thesis, Universitat Oberta de Catalunya, Barcelona 2021.
Rabadia P., Valli C., Ibrahim A., Baig Z., *Analysis of attempted intrusions: intelligence gathered from ssh honeypots* [w:] *The 15th Australian Digital Forensics Conference*, Perth 2017.
Sadique F., Sengupta S., *Analysis of attacker behavior in compromised hosts during command and control* [w:] *ICC 2021 – IEEE International Conference on Communications*, Montreal 2021.
Satria E., Huda T.P.S., Iqbal M., Sarjana F., *The investigation on cowrie honeypot logs in establishing rule signature snort* [w:] *International Conference on Agricultural Technology, Engineering, and Environmental Sciences (ICATES)*, Banda Aceh 2020.
Setianto F. i in., *Gpt-2c: A gpt-2 parser for cowrie honeypot logs*, 2021, https://arxiv.org/abs/2109.06595 [dostęp: 7.01.2023].
Ukkonen E., *On-line construction of suffix trees*, „Algorithmica" 1995, t. 14, nr 3.
Wang B., Chen J., Yu C., *An ai-powered network threat detection system*, „IEEE Access" 2022, t. 10.

Application of suffix trees to efficient presentation of honeypot registered sessions

Abstract

The article presents a prototype of a system for analyzing data from a honeypot network. A special attention is paid to finding similarities in the collected ssh sessions. The algorithm proposed looks for generalized patterns in the session using suffix trees. The patterns can be used for a convenient presentation of the displayed sessions and for searching. The examples of analysis carried out with the help of the algorithm are presented.

Key words: honeypots, malware, session analysis, suffix trees