Raport
Instytutu Automatyki i Informatyki Stosowanej
Politechniki Warszawskiej

Wielotokenowe metody wytwarzania sygnatur

Adam Kozakiewicz

E-mail: ***akozakie@elka.pw.edu.pl***

Raport nr: *12-20*

Warszawa, *12.12.2012*

Raport wykonany w ramach grantu dziekańskiego p.t. „Wydajna generacja i weryfikacja sygnatur zagrożeń aktywnych", porozumienie z dn. 30.06.2012.

Spis treści

Abstrakt: Główną część raportu stanowi przegląd literatury dotyczącej metod automatycznego wytwarzania sygnatur zagrożeń aktywnych, wzbogacony o próbę uporządkowania i ustrukturalizowania zdobytej wiedzy. Podjęto próbę wprowadzenia ścisłej systematyki metod wytwarzania sygnatur. Przedstawiono wykorzystywane w praktyce sposoby rozwiązywania problemów wspólnych dla różnych metod. Zaprezentowano także przegląd proponowanych w literaturze rozwiązań, koncentrujący się na tych, które zostały faktycznie zaimplementowane i dla których dostępne są wyniki praktycznych testów. Na zakończenie przedstawiono badane obecnie w ramach grantu „Wydajna generacja i weryfikacja sygnatur zagrożeń aktywnych" rozszerzenia istniejących metod.

Słowa kluczowe: bezpieczeństwo sieci komputerowych, sniffing, intrusion detection / prevention, automatyczna generacja sygnatur, algorytmy tekstowe

1. Wprowadzenie

Jedną z podstawowych metod obrony przed zagrożeniami sieciowymi propagującymi się aktywnie jest ich wykrywanie i neutralizacja przez systemy IDS/IPS. Aby zidentyfikować zagrożenie, system taki potrzebuje jednak pewnego wzorca, który pozwoli mu zidentyfikować złośliwe komunikaty. Wzorzec taki to **sygnatura** zagrożenia.

Możliwe i od lat praktykowane jest ręczne tworzenie sygnatur na podstawie analizy zachowania złapanego złośliwego oprogramowania. Niestety, w obliczu propagujących się automatycznie poprzez sieć robaków internetowych, jest to rozwiązanie o całe rzędy wielkości zbyt wolne. Czas przygotowania dobrej sygnatury bez wstępnej wiedzy o naturze zagrożenia można mierzyć w godzinach, a niekiedy dniach. Tymczasem, jak wynika z badań [1], czas, w którym propagacjado systemów IDS/IPS nowych sygnatur może powstrzymać epidemię, mierzony jest zaledwie w minutach. Po kilku godzinach znacznie ponad 90\% podatnych komputerów jest już zarażonych i sygnatura pomaga co najwyżej chronić nieliczne ocalałe, ograniczać ruch generowany przez robaka i ewentualnie pomagać w wyszukiwaniu zarażonych. Sygnatury należy zatem tworzyć bardzo szybko i bezzwłocznie wdrażać do ochrony sieci. Jedyną szansą na osiągnięcie tego celu jest wdrażanie automatycznych systemów wytwarzania sygnatur.

W takim zastosowaniu kluczowa jest jakość powstających sygnatur. Z praktycznego punktu widzenia na ocenę jakości sygnatur wpływają trzy różne aspekty pracy systemów wykrywania włamań:

- selektywność (wiarygodność),
- wrażliwość (skuteczność),
- wydajność.

Pierwsza z wymienionych cech narzuca wybór sygnatur gwarantujących niski poziom fałszywych alarmów. Druga z cech nakazuje tworzenie sygnatur, które umożliwiają skuteczne wykrycie występujących w sieci robaków. Na szczęście nie muszą to być pojedyncze sygnatury, możliwe i sensowne jest używanie całych ich zbiorów. Zbiory te nie mogą być jednak zbyt liczne, w związku z cechą trzecią – porównywanie ruchu sieciowego ze zbyt dużą liczbą sygnatur jest zbyt kosztowne czasowo.

Automatyczne tworzenie sygnatur ma jednak także inne zastosowania:

- Grupowanie podobnych przepływów upraszczające ich klasyfikację. Jeśli sygnatura jest dobrej jakości, to można ją opisać. W ten sposób identyfikuje się jednoznacznie zjawisko, które sygnatura wykrywa. Opis określa też, czy jest to atak, czy szum. W ten sposób nie ma potrzeby odrębnej klasyfikacji każdego zarejestrowanego przepływu, co jest w sposób oczywisty niewykonalne.
- Wydobywanie istotnych z punktu widzenia dalszej analizy fragmentów przepływów. Próbki zawartości, pojawiające się w sygnaturach, stanowią często bardzo cenną informację na temat sposobu działania robaka lub innego zagrożenia i mogą stanowić punkt wyjścia do dalszej analizy.

- Ręczne wdrożenia. Nawet, jeśli automatyczne wdrażanie sygnatur uznać za zbyt ryzykowne, mogą one stanowić doskonały punkt wyjścia do ręcznej pracy nad sygnaturą, która będzie polecana użytkownikom do wdrożenia.

Jak widać, systemy tego rodzaju są bardzo istotnymi i użytecznymi narzędziami bezpieczeństwa sieci komputerowych. Mimo to i pomimo wielu lat prac, nadal są dość dalekie od doskonałości.

W niniejszym raporcie przedstawiono najważniejsze informacje dotyczące zagadnień konstrukcyjnych takich systemów. Rozdział 2 przedstawia autorską propozycję klasyfikacji metod generacji sygnatur, wyjaśniając przy okazji na ogólnym poziomie zasady działania systemów różnych typów, ich podobieństwa i różnice. Rozdział 4 zawiera przegląd udokumentowanych w literaturze działających systemów sygnaturowych, skupiający się na szczególnie interesujących z punktu widzenia dalszych badań metodach wielotokenowych. Najpierw jednak, aby uniknąć powtórzeń lub nieporządku w rozdziale 4, rozdział 3 omawia kilka problemów wspólnych, z którymi muszą się mierzyć autorzy systemów tej klasy, w szczególności problemy wyodrębniania tokenów z przepływu, klasteryzacji sygnatur lub przepływów, czy też zbierania i organizacji przetwarzanych danych. Ostatecznie w rozdziale 5 przedstawiono w zarysie kierunki badań prowadzonych przez autora reportu w ramach rozwoju opisywanych metod sygnaturowych.

2. Klasyfikacja metod wytwarzania sygnatur

Sygnatury zagrożeń można dzielić na wiele sposobów. W niniejszym podrozdziale przedstawiono propozycję klasyfikacji sygnatur i systemów sygnaturowych.

2.1. Podział pod względem trybu dopasowania

Najbardziej ogólnym kryterium klasyfikacji systemów sygnaturowych, jest podział na klasy zależnie od sposobu przeprowadzania klasyfikacji przepływów na podstawie danej sygnatury. Klasyfikacja taka w głównej mierze dotyczy oczywiście formy samej sygnatury, jednak ten sam podział można zastosować do całych systemów – sposób dopasowania wynikowych sygnatur w oczywisty sposób wpływa na konstrukcję algorytmów ich wytwarzania. Wyróżniamy zatem sygnatury:

- **Ścisłe** – zawartość przepływu porównywana jest z całością sygnatury, w sposób odpowiedni do rodzaju używanego w sygnaturze wzorca, co daje jednoznaczną klasyfikację. Przebieg w jakikolwiek sposób odbiegający od sygnatury jest automatycznie klasyfikowany jako niepasujący. Grupa ta obejmuje np. wszelkiego rodzaju sygnatury, które można wyrazić w formie wyrażeń regularnych, w tym proste podciągi.
- **Wariantowe** – porównanie przepływu z sygnaturą jest wielofazowe, przy czym w każdej fazie wykorzystywany jest inny fragment sygnatury, a jego wybór może zależeć od przebiegu poprzednich faz. Ostatecznie zatem nie jest wymagana zgodność przepływu ze wszystkimi elementami sygnatury, która zresztą może zawierać nawet elementy sprzeczne, o ile występują one w różnych ścieżkach przetwarzania. Prostym przykładem sygnatury tego typu mogłyby być sygnatury oparte na drzewach decyzyjnych. Sygnatury takie co do zasady mogą zostać zdekomponowane na zbiór niezależnych sygnatur ścisłych, pokrywających dokładnie tę samą przestrzeń przepływów.
- **Rozmyte** – podobnie, jak w przypadku sygnatur wariantowych, nie jest wymagane dopasowanie przepływu do całości sygnatury, decyzja podejmowana jest jednak w sposób bardziej złożony, niż seria prostych dopasowań. Możliwe jest stosowanie punktowania i progów akceptacji, mechanizmów typu „czarnej skrzynki” i temu podobnych rozwiązań. Do tej grupy należy większość technik automatycznej klasyfikacji opartych na uczeniu maszyn, z wyjątkiem prostych drzew decyzyjnych, które – jak już wspomniano – należą do grupy sygnatur wariantowych. Co do zasady, nie jest możliwe proste zastąpienie takiej sygnatury zbiorem sygnatur ścisłych, choć w konkretnych przypadkach może to być możliwe.

Wykorzystywane obecnie w praktyce systemy automatycznego wytwarzania sygnatur tworzą niemal wyłącznie sygnatury ścisłe, choć w literaturze można się zetknąć z dwoma zaimplementowanymi i przetestowanymi systemami generującymi sygnatury rozmyte. Autor nie zetknął się z systemami automatycznego wytwarzania sygnatur wariantowych operującymi bezpośrednio na tekście, podejście to jest jednak naturalne, jeśli przy generacji wykorzystywane są informacje semantyczne.

2.2. Podział ze względu na wykorzystywane cechy

Automatyczna klasyfikacja opiera się zawsze na badaniu pewnych cech klasyfikowanych obiektów. Zazwyczaj więc konieczne jest wprowadzenie dodatkowej fazy wstępnego przetwarzania, w której obiekt jest badany w celu określenia jego cech niezbędnych algorytmowi. W niektórych przypadkach bywa to zadanie o wiele bardziej złożone, niż sama klasyfikacja.

Można zatem wprowadzić podział mechanizmów sygnaturowych pod względem wykorzystywanych do klasyfikacji cech przepływów. Podział ten jest stosunkowo prosty, ale bardzo istotny – własności mechanizmów z różnych grup znacznie się różnią.

- **Syntaktyczne** – systemy traktujące przepływ jako ogólny ciąg znaków.
  - ***Tekstowe*** – badana jest bezpośrednio zawartość przepływu, traktowana jako ciąg bajtów. Przetwarzanie wstępne właściwie nie jest potrzebne, a do klasyfikacji używane są proste algorytmy tekstowe (patrz np. [2]), sprawdzające zgodność zawartości z wyrażeniem regularnym, czy też występowanie w niej zadanego podciągu.
  - ***Mapujące*** – badane są cechy wyekstrahowane z zawartości przepływu, np. o charakterze skrótów czy próbek. W pewnym sensie zatem przepływy są przed rozpoczęciem właściwej klasyfikacji mapowane do innej przestrzeni – przestrzeni cech.
- **Semantyczne** – systemy uwzględniające semantykę protokołów zidentyfikowanych w przepływie.
  - ***Parsujące*** – przepływy są wstępnie przetwarzane z wykorzystaniem znajomości semantyki wykorzystywanego w nich protokołu. Sygnatury tej grupy są mniej ogólne, mogą też być wytwarzane jedynie dla znanych protokołów. W zamian jednak dostępne są nowe możliwości, np. lokalizowanie wzorców wchodzących w skład sygnatury nie względem jej początku lub końca, ale w konkretnych polach wiadomości, bez względu na ich położenie i uporządkowanie. Mimo daleko idącego przetwarzania, klasyfikacja także w tym przypadku odbywa się na podstawie zawartości przepływu.
  - ***Behawioralne*** – przepływy klasyfikowane są nie według swojej zawartości, a według jej skutków dla aplikacji. Wstępne przetwarzanie jest bardzo złożone i obejmuje modelowanie zmian stanu aplikacji przyjmującej dane połączenie.

Rozróżnienie pomiędzy sygnaturami syntaktycznymi i semantycznymi jest bardzo istotne z punktu widzenia obszaru zastosowań i elastyczności proponowanych rozwiązań. W prowadzonych przez autora badaniach nacisk położony jest zdecydowanie na sygnatury syntaktyczne. Ich zaletą z badawczego punktu widzenia jest ich uniwersalność. Odpowiednie modele semantyczne muszą być opracowywane osobno dla każdego protokołu, co zawęża zakres działania mechanizmu. Co więcej, otoczenie sieciowe jest dodatkową zmienną, występującą jedynie w sygnaturach semantycznych: zmiany w kolejnych wersjach protokołów, czy też zmiany ich popularności wpływają bezpośrednio na skuteczność systemu semantycznego, podczas gdy dla systemu syntaktycznego są bez znaczenia, wytwarzane będą inne sygnatury, ale mechanizm nie wymaga zmian. Mechanizmy syntaktyczne dezaktualizują się jedynie w przypadku zmian w występujących w sieci zagrożeniach powodujących naruszenie przyjętych w systemie założeń. Oczywiście działanie ich uniemożliwia

także tunelowanie atakowanych protokołów w kanałach szyfrowanych, jednak dotyczy to w takim samym stopniu systemów semantycznych.

2.2.1. Rodzaje sygnatur tekstowych

W związku z koncentracją na systemach syntaktycznych, wskazane jest uważniejsze przyjrzenie się ich cechom. Wprowadzanie dalej idącej klasyfikacji systemów mapujących wydaje się w obecnej chwili nieuzasadnione – jest ich zbyt mało, abo klasyfikacja taka była użyteczna. W wypadku sygnatur tekstowych jest jednak inaczej. Opracowano (teoretycznie lub praktycznie) wiele takich systemów, znacznie różniących się pod względem postaci sygnatury. Większość znanych systemów sygnaturowych to właśnie mechanizmy tekstowe. Traktowanie ich jako jednej, spójnej grupy byłoby daleko idącym uproszczeniem. Można zatem zaproponować następujący podział tej klasy metod:

- **Jednotokenowe** – sygnatura składa się z pojedynczej próbki, której wystąpienie w ramach przepływu jest równoznaczne z dopasowaniem sygnatury.
- **Wielotokenowe –** sygnatura składa się z wielu tokenów, czyli fragmentów zawartości pakietu, które mogą być dopasowywane niezależnie lub zależnie.

Sygnatury jednotokenowe to jednolita, dobrze zdefiniowana grupa. Sygnatury wielotokenowe można natomiast podzielić według trzech ortogonalnych kryteriów:

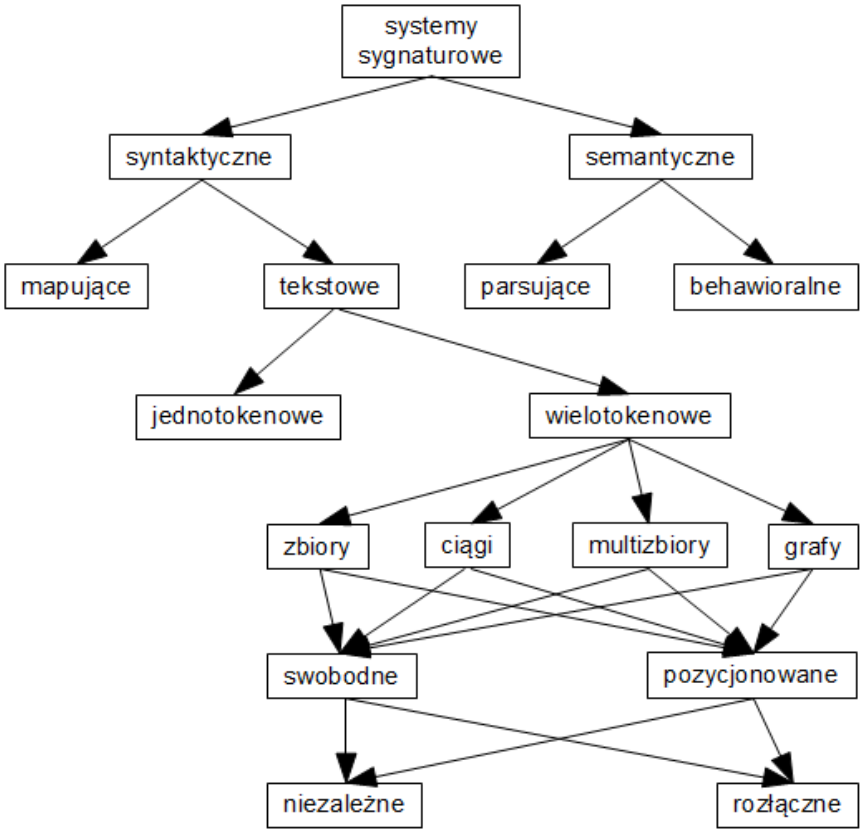
- Sposób zgrupowania tokenów w sygnaturze:
  - ***zbiory tokenów*** – nieuporządkowana lista próbek, których łączne wystąpienie w przepływie oznacza dopasowanie,
  - ***ciągi tokenów*** – uporządkowana lista próbek, które muszą wystąpić w określonej kolejności; ten typ sygnatur szczególnie łatwo daje się wyrazić w formie wyrażeń regularnych, czy też podsekwencji (podciągów z dziurami),
  - ***multizbiory tokenów*** – nieuporządkowane listy próbek wraz z licznikami, specyfikującymi ile razy ma wystąpić dany token, aby można było stwierdzić dopasowanie,
  - ***grafy tokenów*** – zbiory tokenów, w których między poszczególnymi problemami mogą, ale nie muszą występować ograniczenia następstwa.
- Ograniczenia pozycji tokenu:
  - ***swobodne*** – każdy z tokenów może występować w przepływie w dowolnym miejscu,
  - ***pozycjonowane*** – dla każdego z tokenów należących do sygnatury podany jest zakres pozycji, na jakich może występować w przepływie.

  Pozycjonowanie może z łatwością być wykorzystane w zbiorach, ciągach lub multizbiorach tokenów. Utworzenie grafów tokenów pozycjonowanych wymaga wprowadzenia dodatkowych pseudotokenów oznaczających początek i koniec przepływu oraz metryk krawędzi, wyznaczających zakresy dopuszczalnych odległości między tokenami. Jest to

najbardziej zaawansowany typ sygnatury wielotokenowej, jednak w literaturze brak śladu prób uzyskania tak skomplikowanych, choć zawierających maksimum informacji sygnatur.

- Wzajemne ograniczenia dopasowania tokenów:
  - ***rozłączne*** – dopasowane wystąpienia tokenów w przepływie nie mogą mieć części wspólnych,
  - ***niezależne*** **–** dopasowane tokeny w przepływie mogą mieć części wspólne.

  Procedura dopasowania tokenów niezależnych jest oczywiście znacznie prostsza. W przypadku tokenów rozłącznych procedura może uwzględniać wszystkie wystąpienia danego tokenu w przepływie, co skutkuje dużą złożonością dopasowania, lub ograniczać się do znalezienia pierwszego wystąpienia, co może uzależniać możliwość dopasowania sygnatury od kolejności rozpatrywania tokenów.



Rysunek 1: Klasyfikacja sygnatur

2.3. Własności sygnatur różnych typów

Najistotniejszą cechą sygnatury jest jej zdolność wykrywania zagrożeń, które opisuje. Zdolność ta zależy jednak nie tylko od samej sygnatury. Zagrożenia są zróżnicowane pod względem kierunku propagacji i zmienności, co ma duży wpływ na możliwość skutecznego wykrywania.

Kierunek propagacji nie ma w zasadzie wpływu na skuteczność istniejących sygnatur, może natomiast znacznie utrudnić ich wytworzenie, uzależnione od zdobycia dostatecznej liczby próbek, aby automat tworzący sygnatury miał szansę zareagować. Najłatwiejszym przypadkiem jest prosta propagacja skanująca, czy to losująca adresy IP, czy też atakująca je po kolei. Robaki wolno propagujące, które sztucznie ograniczają częstość prób propagacji są trudniejsze do wykrycia, jednak jednocześnie mniej groźne, gdyż rozprzestrzeniają się powoli. Najgorszym możliwym przypadkiem są robaki działające na podstawie listy celów, które w zasadzie nigdy nie trafiają do honeypotów. Bez względu jednak na to, z którym typem propagacji mamy do czynienia, skuteczność sygnatur będzie taka sama, kiedy już uda się zdobyć odpowiedni zbiór podejrzany i wytworzyć sygnaturę.

Zróżnicowanie robaków pod względem zmienności jest o wiele istotniejsze z punktu widzenia ich wykrywania. W literaturze najczęściej wspominane są następujące stopnie zmienności zagrożeń:

- **Zagrożenia monomorficzne** to najprostszy typ zagrożeń – robaki, które propagują się w niezmienionej formie na różne adresy IP. Zagrożenia te są stosunkowo proste do wykrycia, z uwagi na duże podobieństwo poszczególnych wystąpień.
- **Zagrożenia oligomorficzne** (inaczej **wielowariantowe**) nie różnią się istotnie od monomorficznych. Są to robaki, które – czy to w ramach prac nad ich kodem, czy dostosowania do różnych celów – wypuszczono w kilku różniących się, lecz bardzo podobnych wersjach, lub też robaki, które potrafią wykonywać kilka różnych wariantów ataku.
- **Zagrożenia polimorficzne (*polimorfizm treści*)** – ten przypadek jest znacznie bardziej skomplikowany. Większość zawartości przepływu jest w tym przypadku losowa. Istnieją jednak fragmenty, w których występuje właściwy kod robaka, lub – jeśli ten jest pobierany osobnym kanałem – kod exploita i downloadera. Najgorszym przypadkiem jest wariant, w którym zawartość pakietu jest uzależniona od czasu i stosunkowo wolno się zmienia. Pojawi się wtedy wiele przepływów o bardzo podobnej zawartości, przy czym podobieństwo będzie występować w znacznej mierze w części polimorficznej. Jeszcze groźniejszy efekt można uzyskać zmieniając całkowicie większość polimorficznej części przepływu, pozostawiając przy tym dość długi wolno zmienny fragment.
- **Zagrożenia polimorficzne (*polimorfizm behawioralny*)** – robak jest w stanie modyfikować nie tylko wolne miejsca w przepływach, ale i własny kod, osiągając te same skutki przy użyciu wielu różnych kombinacji instrukcji.

Tworzenie sygnatur dla zagrożeń mono- i oligomorficznych nie stanowi większego problemu. Skuteczne są w zasadzie wszystkie typy sygnatur, bo nawet proste syntaktyczne sygnatury jednotokenowe mogą w tym przypadku być długie, selektywne i skuteczne. Zmiany występujące w

zagrożeniach oligomorficznych mogą utrudnić stworzenie pojedynczej skutecznej sygnatury, jednak zwykle nie uniemożliwiają tego, zwłaszcza w przypadku zastosowania sygnatur wielotokenowych. Nawet przy użyciu najprostszych sygnatur jednotokenowych można je z łatwością opisać zbiorem sygnatur opisujących poszczególne warianty jak osobne robaki.

Wprowadzenie polimorfizmu jest poważnym utrudnieniem dla sygnatur syntaktycznych, a zwłaszcza jednotokenowych – wygenerowana sygnatura może być zbyt wrażliwa lub mało specyficzna. Dobrze zrealizowany mechanizm polimorficzny zmniejsza skuteczność metod jednotokenowych niemal do zera. Szczególnie trudny jest wspomniany przypadek polimorfizmu ze spowolnioną zmianą całości lub części zmienianej sekcji przepływu. Dla sygnatur jednotokenowych jest to przypadek skrajnie trudny – jest bardzo prawdopodobne, że wygenerowana sygnatura w ogóle pominie fragmenty charakterystyczne dla robaka, skupiając się na długich, pozornie stałych fragmentach polimorficznych. Mechanizmy wielotokenowe są jednak dość odporne na tę formę polimorfizmu. W szczególności jeśli badany zbiór zawiera jakiekolwiek próbki różniące się całkowicie częścią polimorficzną, większość algorytmów wielotokenowych wygeneruje pojedynczą sygnaturę skupiającą się na fragmentach charakterystycznych dla robaka, podczas gdy algorytmy jednotokenowe z dużym prawdopodobieństwem wybiorą jednak wyraźnie dłuższe fragmenty wspólne części polimorficznej, tworząc cały zbiór lepiej dopasowanych do próbek, ale nieskutecznych sygnatur. Sygnatury semantyczne radzą sobie z takim polimorfizmem znacznie lepiej, choć i w tym przypadku jest to zadanie trudniejsze niż zwykle.

Najciekawszym zagadnieniem jest wykrywanie robaków stosujących polimorfizm behawioralny. Jest to problem w dużej mierze teoretyczny – do dziś nie jest znany żaden przypadek występującego w praktyce robaka tego rodzaju. Nie można jednak wykluczyć pojawienia się takich wariantów w przyszłości. Początkowo trudność tego zagadnienia była przedstawiana jako argument za rozwojem metod semantycznych, w szczególności behawioralnych. Szybko jednak okazało się, że przyjmowane początkowo założenie, że ta forma polimorfizmu uniemożliwi całkowicie stosowanie prostych, tekstowych metod wykrywania, jest jednak błędne. Argumenty za taką tezą można znaleźć w większości cytowanych artykułów.

Pełna skuteczność takiego polimorfizmu dotyczy jedynie wirusów – w ich przypadku rzeczywiście często spotykana jest w praktyce i wymusza stosowanie heurystycznych metod semantycznych, jako że jedyną stałą częścią programu wirusa (rozumianego jako ciągu binarnego) są fragmenty występujące w każdym pliku wykonywalnym danego formatu. Robak sieciowy działa jednak w zupełnie innych warunkach. Przesyłany w sieci kod wirusa to jedynie dane zawarte w pakietach, całkowicie niegroźne, dopóki nie uda się doprowadzić do ich wykonania jako kodu wykonywalnego. Kod właściwego robaka może zatem być polimorficzny, ale doprowadzenie do jego wykonania wymaga wykorzystania luki w jakiejś aplikacji, co można zrobić tylko na pewne określone sposoby. To wymaganie oznacza, że przepływ sieciowy, w którym propaguje się robak polimorficzny, ma co najmniej dwa rodzaje stałych fragmentów, które mogą zmieniać się tylko w niewielkim zakresie.

Po pierwsze, złośliwy przepływ musi wprowadzić serwer, z którym się komunikuje, w odpowiedni stan, w którym występuje podatność. Sposób osiągnięcia tego celu może być różny, jednak jakiś fragment pakietu musi mieć ściśle określoną formę, która nadaje się na fragment sygnatury.

Selektywności sygnatur opartych na elementach samego protokołu może wydawać się wątpliwa, jednak kluczowe, często wykorzystywane części protokołu są zwykle dobrze przetestowane i rzadko zawierają luki. Skuteczny exploit wykorzystuje zatem najczęściej luki w rzadszych rodzajach wywołań danego protokołu.

Drugi wrażliwy punkt robaka jest związany ze sposobem eksploatacji luki. Aby doprowadzić do wykoania swojego kodu, robak musi wymusić skok do odpowiedniego miejsca w jednej ze standardowych, możliwie uniwersalnie spotykanych bibliotek, tak aby znajdujący się tam kod skonsumował podłożone wartości i oddał mu kontrolę. Lista adresów, pod którymi znajduje się na większości atakowanych komputerów pożądany kod, jest krótka – mowa o zaledwie kilku adresach. Adresy te są wykorzystywane we wczesnej fazie ataku, kiedy kod robaka jeszcze nie działa. Muszą zatem pojawić się w pakiecie w formie jawnej. Jeśli odpowiedni adres jest tylko jeden, to stanowi gotowy fragment skutecznej sygnatury. Większa liczba adresów skutkuje co najwyżej wytworzeniem kilku różnych sygnatur.

Poza wspomnianymi, przepływ generowany przez robaka może zawierać więcej stałych fragmentów. Dodatkowe mogą pojawić się w wyniku niedoskonałości samego silnika polimorficznego. Podsumowując zatem, wykrywanie robaków jest możliwe przy użyciu metod tekstowych nawet w przypadku silnie polimorficznych zagrożeń. Niemniej wybór typu sygnatury nie jest bez znaczenia. W szczególności z powyższego opisu widać, że bardzo charakterystyczny dla danego robaka może być bardzo krótki ciąg bajtów – pojedynczy adres w pamięci. Odpowiednio wrażliwe sygnatury muszą zatem być w stanie uwzględniać takie krótkie fragmenty. Odporność na polimorfizm behawioralny jest więc wśród sygnatur syntaktycznych osiągalna jedynie dla algorytmów wielotokenowych, lub ewentualnie odpowiednio skonstruowanych algorytmów mapujących. Nie jest natomiast jasne, który typ sygnatur z tej grupy jest szczególnie godny polecenia. Odpowiednie analizy stanowią znaczną część treści dostępnych w literaturze artykułów. Przedstawiane w nich konkluzje nie pozwalają jednak na jednoznaczny wybór. Nowoczesne metody z tej grupy tworzą według publikowanych testów sygnatury w zasadzie wystarczająco selektywne i wrażliwe, oscylujące na pograniczu całkowitego braku fałszywych alarmów i pominięć. Kluczowe przy wyborze rozwiązania do zastosowania w praktyce wydają się zatem inne cechy: wydajność procesu wytwarzania, odporność na szum w zbiorze podejrzanym oraz, co chyba najważniejsze, praktyczne walory powstających sygnatur, czyli łatwość ich wyrażenia jako reguł w typowych systemach IDS oraz wydajność ich dopasowania.

Istotną uwagą końcową jest obserwacja, że do testowania mechanizmów sygnaturowych używa się obecnie w niemal wszystkich pracach wyłącznie sztucznie wygenerowanych danych – najczęściej powstałych z przetworzenia zarejestrowanych przepływów znanych robaków. Celem jest wprowadzenie do testów symulowanego efektu polimorfizmu behawioralnego. Większość treści przepływu zastępuje się zatem losowymi ciągami bajtów, pozostawiając bez zmian jedynie fragmenty kluczowe, które musiałyby wystąpić także gdyby robak był polimorficzny. Powodem takiego postępowania jest brak zarejestrowanych próbek takich ataków występujących w rzeczywistości.

3. Problemy wspólne dla mechanizmów różnych typów

Niniejszy rozdział omawia problemy występujące w wielu bądź wszystkich rozwiązaniach wytwarzających sygnatury oraz stosowane w praktyce sposoby rozwiązywania tych problemów. W szczególności omówione zostaną problemy tokenizacji i klasteryzacji, a także uniwersalny problem pozyskiwania danych do przetwarzania, czyli automatyczna klasyfikacja i rejestracja ruchu.

3.1. Podział przepływów na tokeny

Zadanie wydobycia doskonałej wielotokenowej sygnatury bezpośrednio ze zbioru podejrzanego jest problemem NP-trudnym, a przez to praktycznie nierozwiązywalnym, nawet dla niewielkich zbiorów. W praktyce zatem zadanie to często dekomponowane jest na dwie fazy. W pierwszym kroku ze zbioru przepływów wydobywane są często występujące tokeny, będące dobrymi kandydatami na fragmenty sygnatur. Następnie, w drugiej fazie wybiera się zestaw nadający się do zastosowania w roli sygnatury.

W literaturze spotykane są dwa znacząco różne podejścia do tworzenia tokenów. W systemie Autograph [3] stosowany jest automatyczny, pełny podział na tokeny na podstawie skrótów Rabina. Pozostałe systemy stosują wyszukiwanie wspólnych podciągów. W każdym przypadku duże znaczenie ma dostępność zbioru normalnego, czyli zarejestrowanych przepływów niezłośliwych, stanowiących wzorzec prawidłowego ruchu. Tokeny często występujące w tym zbiorze nie są interesujące, gdyż nie pomagają wykryć zagrożenia.

Podział na tokeny według kodu Rabina jest prostym algorytmem probabilistycznym. Przepływ jest przetwarzany na ciąg skrótów Rabina [4] i dzielony w miejscach wyznaczonych przez wartości tych skrótów. Podział występuje w miejscach wystąpienia skrótów, które są równe modulo a pewnej założonej wartości. Poprzez odpowiedni dobór parametru a można regulować oczekiwaną długość tokena. W praktyce minimalna długość tokena jest dodatkowo sztywno ograniczana, aby uniknąć tworzenia tokenów bardzo krótkich. Realizacja tego ograniczenia jest bardzo prosta – po dokonaniu podziału kilka następnych wartości skrótu jest ignorowanych, podział zatem nie nastąpi nawet jeśli są równe modulo a.

Mechanizm taki jest wprawdzie absolutnie heurystyczny, jednak nieporównanie lepszy od podziału na tokeny stałej długości, czy losowego. Powiązanie z zawartością przepływu poprzez skróty Rabina zapewnia nieprzypadkowe miejsca podziału, co obniża ryzyko rozdzielenia znaczących fragmentów zawartości, które powinny znaleźć się w sygnaturze. Niemniej granice powstających tokenów nie są w żaden sposób powiązane z krańcami rzeczywistych stałych fragmentów zagrożeń, powstające z nich sygnatury nie są zatem optymalne.

Znacznie lepiej umotywowanym, choć bardziej kosztownym sposobem podziału na tokeny jest wyszukiwanie fragmentów wspólnych dla wielu przepływów złośliwych. Zadanie wyszukania odpowiednio wielu podciągów wspólnych dla dostatecznie dużej grupy podejrzanych przepływów jest znane, dobrze zdefiniowane i ma wiele możliwych rozwiązań. W systemie Polygraph [5] wykorzystywany jest znany algorytm wyszukiwania najdłuższego wspólnego podciągu w K z n

próbek. Został on zmodyfikowany tak, aby zamiast najdłuższego zwracał wszystkie podciągi występujące w odpowiednio wielu próbkach. Jest to algorytm o złożoności liniowej. Aby uzyskany zbiór nadawał się do użycia, konieczne jest jego dodatkowe przetworzenie w celu usunięcia ciągów nieunikalnych, będących podciągami innych. Zastosowana do tego procedura przycinania pozostawia jedynie możliwie długie fragmenty unikalne. Algorytmy Hamsa [6] i Nebula [7] rozwiązuja problem podziału inaczej, wykorzystując algorytm oparty na tablicach sufiksowych. W obu podejściach uzyskiwane zbiory tokenów dobrze charakteryzują zbiór podejrzany.

3.2. Grupowanie obiektów w klastry

Generowane przez rozważane systemy sygnatury nie zawsze są zadowalającej jakości. Dotyczy to w szczególności systemów tworzących dla danego zbioru podejrzanego wiele różnych sygnatur. Częstym zjawiskiem jest wtedy powstawanie wielu sygnatur podobnych, z których każda z osobna jest zbyt selektywna. Pogarsza to oczywiście praktyczną użyteczność powstałego zbioru – sprawdzanie dużej liczby podobnych sygnatur obniża wydajność systemów typu IDS, więcej niepotrzebnej pracy wymaga też ręczne etykietowanie sygnatur, które jest niezbędne w celu analizy zarejestrowanych zjawisk.

Z drugiej strony, systemy wytwarzające tylko jedną sygnaturę dla całego zbioru podejrzanego mają niższą odporność na skomplikowane zbiory wejściowe – kiedy zbiór podejrzany zawiera wystąpienia kilku różnych robaków i dodatkowy szum, powstająca sygnatura obejmująca wszystkie przypadki będzie oczywiście bardzo mało wartościowa.

Oba te problemy można ograniczyć przez zastosowanie metod grupowania podobnych obiektów w klastry, nazywanego automatyczną klasteryzacją. W przypadku problemu zbyt selektywnych sygnatur, klasteryzację przeprowadza się na samych sygnaturach. Dla powstałego klastra podobnych sygnatur można wtedy łatwo wygenerować sygnaturę łączną, uogólniającą sygnatury składowe. W przypadku sygnatur zbyt ogólnych klasteryzację stosuje się natomiast do zbioru podejrzanego, dzieląc go na grupy podobnych przepływów i pozbywając się w ten sposób szumu.

Istnieje wiele metod klasteryzacji. W działających systemach wytwarzania sygnatur wykorzystano dotychczas trzy.

- **Klasteryzacja hierarchiczna** wykorzystywana jest w systemie Polygraph [5] w celu podziału zbioru podejrzanego. Wbrew opiniom wygłaszanym w artykule, algorytm wydaje się skrajnie nieefektywny. Przebiega on następująco:
    1. Niech zbiór podejrzany składa się z s przepływów. Utwórz początkowy zbiór s klastrów jednoprzepływowych i wygeneruj dla nich sygnatury.
    2. Dla wszystkich par klastrów wygeneruj sygnatury, jakie powstałyby, gdyby je połączyć.
    3. Spośród sygnatur utworzonych dla par klastrów wybierz, poprzez zbadanie na zbiorze normalnym, tę, która ma najniższy współczynnik fałszywych alarmów.
    4. Jeśli współczynnik fałszywych alarmów przekracza akceptowalny poziom (który jest parametrem algorytmu), STOP. Wynikiem algorytmu jest aktualny zbiór klastrów.

5. Jeśli powstały klaster obejmuje wszystkie próbki, to jest on jedynym wynikiem algorytmu, STOP.
6. Jeśli żaden warunków (5, 6) nie zachodzi, dodaj nowy klaster do puli, usuwając z niej jego klastry składowe.
7. Wygeneruj sygnatury, jakie powstałyby z połączenia nowego klastra ze wszystkimi już istniejącymi, po czym przejdź do kroku 3.

Jak uczciwie przyznają twórcy, taki algorytm klasteryzacji ma złożoność kwadratową. Zdaniem autorów jest to szybkie i w pełni zadowalające rozwiązanie. W rzeczywistości wiele zależy od sposobu implementacji i rozmiarów zbioru normalnego. Trzeba też zauważyć, że algorytm ten wymaga wytworzenia $O(n^2)$ sygnatur, gdyż każdy możliwy do stworzenia klaster musi uzyskać sygnaturę i ocenę jej jakości. Takie podejście sprawdza się w systemie Polygraph wyłącznie dlatego, że tworzenie sygnatur jest w tym systemie operacją bardzo prostą.

Klasteryzacja hierarchiczna jest zatem sensowna wyłącznie w przypadku mechanizmów wytwarzania sygnatur, w których znaczna część pracy wykonywana jest w fazie przygotowania danych, a więc w algorytmach wielotokenowych, w których tokeny są wyodrębniane i oceniane w fazie wstępnej. W tym przypadku możliwe jest ponowne wykorzystanie informacji o powiązaniu tokenów z przepływami, w których wystąpiły, co sprawia, że zarówno wytworzenie sygnatury, czyli wybór tokenów, jak i ich ocena, czyli zbadanie częstości występowania ich kombinacji w przepływach normalnych, są operacjami o bardzo niskim koszcie obliczeniowym. Stosowanie tego podejścia jest natomiast niepraktyczne, gdy tworzenie sygnatury i jej ocena wymaga stosowania złożonych algorytmów tekstowych.

- **DBSCAN** [8] jest algorytmem klasteryzacji ogólnego zastosowania, wykorzystywanym (w lekko zmodyfikowanej wersji) w systemie ARAKIS. Jest to algorytm całkowicie innego typu: klasteryzacja hierarchiczna całkowicie abstrahuje od klastrowanych obiektów, oceniając wyłącznie jakość powstających klastrów, podczas gdy DBSCAN jest algorytmem przeznaczonym do grupowania obiektów podobnych, niezależnie od użyteczności powstającego klastra.

  Stosowanie algorytmu wymaga zdefiniowania sposobu pomiaru stopnia podobieństwa obiektów. Istnieje wiele różnych miar podobieństwa stosowanych dla danych tekstowych. Wybór odpowiedniej jest istotny, gdyż decyduje o użyteczności wyników algorytmu. Obiektami klasteryzacji w rozważanym zastosowaniu mogą być np. przepływy, tokeny, czy sygnatury. W systemie ARAKIS grupowane są sygnatury, które mają postać pojedynczych, długich ciągów znaków, stanowiących fragmenty przepływów. Naturalną miarą podobieństwa jest w tym przypadku odległość edycyjna. Wybrano stosunkowo prosty wariant tej odległości: metrykę Levenshteina [9]. Metryka ta uwzględnia jedynie operacje wstawiania i usuwania bajtów. Biorąc pod uwagę fakt, że w przypadku robaków najbardziej prawdopodobną modyfikacją zawartości jest podmiana bajtów lub całych fragmentów tekstu, być może korzystniejszym wariantem byłoby wykorzystanie odległości Damerau-Levenshteina [10], w której zamiana bajtu na inny jest trzecią dozwoloną pojedynczą

operacją, przez co zmiany takie mają mniejszy wpływ na odległość, niż w odległości Levenshteina, gdzie reprezentowane są przez dwie operacje (usunięcie i wstawienie).

Klastry tworzone algorytmem DBSCAN posiadają wyróżnione rdzenie – wybrane obiekty podobne do wystarczająco wielu innych. Klaster jest tworzony z obiektów, których odległość od jego rdzenia jest mniejsza od pewnej wartości progowej. Tworzenie klastrów wykorzystuje prosty algorytm zachłanny, w którym nowe klastry tworzone są natychmiast, kiedy uda się zebrać odpowiednią ilość podobnych obiektów, które nie trafiły do istniejących już klastrów. Dzięki temu jest to algorytm dość szybki i skuteczny, jednak jakość tworzonych klastrów nie zawsze jest zadowalająca.

Korzystną cechą algorytmu DBSCAN w niektórych zastosowaniach jest trwałość klastrów, pozwalająca wiązać z nimi użyteczne metadane. Algorytm może być stosowany przyrostowo, bez potrzeby ponownej klasteryzacji całego zbioru obiektów – obiekty już klastrowane pozostaną w istniejących klastrach, a nowe zostaną do nich dopisane. Powstanie nowego klastra jest zatem istotną informacją i oznacza pojawienie się całej grupy podobnych do siebie obiektów, których nie można dopasować do istniejących klastrów.

- **Klasteryzacja grafowa** wykorzystywana jest w systemie Nebula do wstępnej klasteryzacji zbioru podejrzanego. Algorytm bazuje, podobnie jak DBSCAN, na odległości między obiektami, przy czym w systemie Nebula zamiast prostej odległości decyzyjnej wykorzystywane są skróty spamsum [11]. Zasada działania algorytmu jest dość podobna do DBSCAN, choć w klastrach nie wyróżnia się rdzeni. Każdy kolejny obiekt porównywany jest z poprzednikami. Jeśli istnieją już wystarczająco podobne obiekty (próg podobieństwa jest parametrem algorytmu), to nowy obiekt jest z nimi łączony.

  Dołączenie nowego obiektu prowadzi do utworzenia nowego klastra, kiedy łączy się on tylko z obiektami „samotnymi”, nie należącymi do klastrów. Połączenie z obiektami należącymi do jednego klastra powiększa ten klaster o nowy obiekt, jeśli natomiast podobne są obiekty należące do więcej, niż jednego klastra, dochodzi do scalenia sąsiednich klastrów przez zamknięcie luki między nimi. Autorzy przedstawiają algorytm w interpretacji grafowej, gdzie odpowiada on segmentacji przez usunięcie z pełnego grafu odległości wszystkich krawędzi dłuższych od progu.

  Oczywiście przyrostowy charakter tego algorytmu wymaga stosowania odpowiednich metod przycinania grafu, czyli usuwania starych obiektów. Nie jest to zadanie bardzo trudne, ale wymaga przemyślanego algorytmu – mechaniczne usuwanie wszystkich obiektów starszych od pewnej wartości progowej może prowadzić do rozpadu dobrych w istocie klastrów.

  Trzeba także zwrócić uwagę, że wystarczającym warunkiem przyłączenia nowego obiektu do klastra jest dostatecznie mała odległość od najbliższego jego elementu. Możliwe jest zatem powstawanie „rozciągniętych” klastrów, w których elementy połączone poprzez wiele węzłów pośrednich bardzo różnią się od siebie.

Istnieje wiele innych rozwiązań, należących do różnych rodzin, których wydajność, jak i jakość tworzonych klastrów w niektórych zastosowaniach przewyższają wymienione rozwiązania. Jeśli

zatem przy opracowywaniu nowego algorytmu sygnaturowego pojawi się problem klasteryzacji,, należałoby rozważyć wiele algorytmów i dokonać odpowiedniego wyboru. Znaczenie klasteryzacji jako kroku wstępnego lub końcowego w procesie tworzenia sygnatur wyraźnie jednak maleje. Nowoczesne systemy wielotokenowe wytwarzają niezbyt liczne sygnatury, nie wymagające dodatkowej klasteryzacji. Są one też odporne na zaszumione i wielorobakowe zbiory podejrzane, dzięki odpowiedniej strukturze algorytmu.

3.3. Dane wejściowe

Algorytmy wytwarzania sygnatur nie działają oczywiście w próżni – niezbędny jest zbiór przepływów, które miałyby opisywać tworzone sygnatury, zwany **zbiorem podejrzanym**. Większość algorytmów wykorzystuje jednak również drugi zbiór wejściowy – zbiór ruchu normalnego, najczęściej nazywany **zbiorem normalnym** lub **zbiorem niezłośliwym**. Niniejszy rozdział opisuje znaczenie i sposób tworzenia obu zbiorów. Problemy tego typu dotyczą wszelkich mechanizmów wytwarzania sygnatur.

3.3.1. Problem selekcji zbioru podejrzanego

Wspólnym elementem architektur opisywanych w większości publikacji z dziedziny jest **klasyfikator przepływów**. Większość publikacji traktuje go niestety jako czarną skrzynkę, ewentualnie proponując pewne prototypowe rozwiązanie, zawsze przy tym zaznaczając, że moduł ten może być wymieniony. Jest to podejście dalece niesatysfakcjonujące, gdyż element ten jest bardzo istotny dla poprawnego działania mechanizmu, a jego skuteczna, wydajna realizacja nie jest wcale zadaniem dużo łatwiejszym od samego wytwarzania sygnatur.

Automatyczne systemy wytwarzania sygnatur nie mają zwykle żadnej możliwości realnej oceny złośliwego charakteru rejestrowanego przepływu. Muszą jednak klasyfikować zbierane przepływy jako złośliwe lub nie. Wprowadzenie do nich jednoznacznych reguł nie ma oczywiście sensu – byłyby to de facto ręcznie przygotowane sygnatury i automatyzacja stałaby się fikcją. Stosowane są zatem heurystyki, najczęściej wykorzystujące znane charakterystyki dotychczas zarejestrowanych epidemii robaków, takie jak:

- stałość portów, co najmniej docelowego,
- występowanie skanowań, tzn. łączenie się tego samego nadawcy z wieloma odbiorcami,
- wykładniczy wzrost liczby przepływów na odpowiednim porcie, jak i liczby nadawców.

Heurystyki takie zapewniają wyodrębnienie grupy przepływów *podejrzanych*. Występują w niej liczne niedokładności: jest zarówno zaszumiona połączeniami poprawnymi, które przypadkiem spełniły warunki heurystyki, jak i niekompletna – w szczególności zapewne pominie pierwsze, bardzo nieliczne wystąpienia nowego robaka. Najistotniejszy jest jednak fakt, że w wyodrębnionym zbiorze odsetek przepływów złośliwych jest istotnie wyższy, niż w całym zbiorze.

Niektóre algorytmy zakładają, że zbiór podejrzany zawiera jedynie próbki *jednego* nieznanego robaka, dopuszczając najwyżej pewien poziom szumu, czyli przepływów niezłośliwych. Założenie to oznacza, że robaki znane muszą być wstępnie odfiltrowane przy użyciu znanych już sygnatur.

Przyjmuje się tym samym, że niemal jednoczesny wybuch więcej niż jednej epidemii jest wysoce nieprawdopodobny. Założenie to jest dość istotne dla konstrukcji samego algorytmu, jeśli jest bowiem słuszne, to pożądane jest uzyskanie pojedynczej sygnatury, która opisze cały zbiór podejrzany. W rzeczywistości często znacznie lepsze może okazać się podejście dające nieco większą, choć najlepiej niezbyt dużą liczbę sygnatur. W ten sposób możliwe jest osiągnięcie lepszej ich jakości. Korzystna jest też możliwość pozostawienia bez sygnatury pewnej niewielkiej części przepływów. W wariancie pesymistycznym może to wprawdzie oznaczać pominięcie niektórych wariantów robaka, bardziej prawdopodobne jednak jest to, że ten różniący się znacznie od reszty ruch to właśnie szum wynikający z niedoskonałości klasyfikatora.

Całkowicie odrębnym przypadkiem jest zastosowanie generatora sygnatur jako narzędzia do automatycznego opisu ruchu zbieranego przez systemy honeypot[1]. Ruch taki jest z definicji w całości podejrzany – decyduje sam fakt skierowania go na nieużywany produkcyjnie adres IP. Niestety, choć może się to wydać zaskakujące, jakość takiego zbioru podejrzanego jest w praktyce daleka od doskonałości. Pewna część ruchu rejestrowanego przez honeypoty to w rzeczywistości szum: poprawne, niezłośliwe próby połączeń wynikające z różnego rodzaju błędów w konfiguracjach. Także i w tym przypadku należy więc traktować zbiór jako zaszumiony. Także ruch o złośliwych zamiarach nie musi być próbą propagacji robaka. Znaczna jego część to różnego rodzaju próby skanowań. W praktyce to ostatnie zastrzeżenie ma ograniczone znaczenie, gdyż skanowania są zwykle podobnie niepożądane, jak same ataki. Jeśli więc skanowanie również da się opisać automatycznie wytworzoną sygnaturą, można uznać je za poprawnie obsłużone – dopiero znacznie później, na etapie ręcznej klasyfikacji incydentów, odpowiedni specjalista może określić daną sygnaturę jako identyfikującą ruch niepożądany, ale niegroźny.

3.3.2. Zbiór normalny i jego przeznaczenie

Większość znanych z literatury rozwiązań wykorzystuje zbiór normalnego ruchu w charakterze punktu odniesienia. Zależnie od sposobu działania algorytmu, jest on co najmniej przydatny, a w skrajnych przypadkach niezbędny. Posiadanie takiego zbioru umożliwia odrzucanie sygnatur niespecyficznych, powodujących znaczną liczbę fałszywych alarmów. W przypadku algorytmów wielotokenowych zbioru można użyć także w fazie tokenizacji, czyli jeszcze przed wygenerowaniem sygnatury, mierząc prawdopodobieństwo wystąpienia w legalnym ruchu poszczególnych tokenów.

Kiedy system pracuje na ruchu produkcyjnym, wykorzystując klasyfikator połączeń, najbardziej naturalne mogłoby się wydawać wykorzystanie w tej roli wybranych połączeń ze zbioru klasyfikowanego jako ruch normalny. Podejście to oczywiście nie ma sensu w przypadku pracy na ruchu rejestrowanym przez honeypoty, gdzie przepływów normalnych właściwie nie ma. Okazuje się jednak, że podejście takie jest niekorzystne nawet w przypadku, gdy jest proste do wykonania.

Główną słabością rozwiązań tworzących ruch normalny równolegle ze złośliwym jest ich podatność na atak zwany normal pool poisoning. W ataku tym napastnik wysyła dużą ilość legalnego ruchu,

1 Honeypot, inaczej pułapka – system, którego przenaczeniem jest bycie atakowanym; emuluje rzeczywiste usługi i rejestruje swoją interakcję z napastnikiem; wobec braku produkcyjnych funkcji można przyjąć założenie, że każde połączenia przychodzące jest albo skutkiem błędu (pomyłka, błędna konfiguracja), albo działania niepożądanego – nieautoryzowanego badania konfiguracji sieci lub próby ataku.

który w większości wykorzystywanych w klasyfikatorach heurystyk nie wzbudzi podejrzeń, przy czym ruch ten celowo zawiera tokeny typowe dla nowego robaka. Zadanie jest na tyle proste, że jego wykonanie można wręcz pozostawić samemu robakowi w fazie przygotowania do propagacji. Skuteczny atak nasyca zbiór normalny przepływami zawierającymi podejrzane fragmenty, przez co wszystkie faktycznie skuteczne sygnatury identyfikujące nowe zagrożenie będą automatycznie odrzucane z uwagi na dużą ilość fałszywych alarmów.

Z tego względu zalecane jest wykorzystywanie w roli zbioru normalnego starszych próbek ruchu, które w efekcie nie powinny jeszcze zawierać żadnych śladów ewentualnej rozpoczynającej się właśnie epidemii. Dobrym rozwiązaniem są zbiory kilkumiesięczne – w tak krótkim czasie nie dojdzie zwykle do dużych zmian w popularności protokołów, dane można uznać za aktualne. Dodatkową korzyścią jest dostępność i jakość takich zbiorów. Odpowiednie zbiory przepływów można pobrać z publicznych źródeł, można też wytworzyć je na własne potrzeby monitorując ruch produkcyjny. Co ważne, można to robić raz na pewien czas (miesiąc, czy kwartał), a uzyskany zbiór można w tym czasie bez trudu, choćby ręcznie, zanonimizować i oczyścić z przepływów, których ujawnienie jest z różnych względów niepożądane.

3.3.3. Rejestracja ruchu

Pozyskanie ruchu, zarówno normalnego, jak i złośliwego, wymaga nagrywania przepływów pakiet po pakiecie (*sniffing*). Niezbędne jest zapisywanie pełnego binarnego zrzutu pakietu – zapis np. samych nagłówków jest bezużyteczny, skoro dopasowania najprawdopodobniej pojawią się w treści pakietu. Standardowo tego rodzaju zrzuty ruchu zapisywane są w formacie PCAP. Istnieje jednak kilka różnych sposobów ich rejestracji.

- **Monitorowanie systemowe** to metoda opierająca się na rejestracji ruchu poprzez jego monitorowanie na poziomie interfejsu. Istnieją gotowe narzędzia do takich celów, z których najpopularniejszym jest tcpdump. Zrzuty na tym poziomie odbywają się jednak najczęściej na poziomie pojedynczych pakietów, nie powiązanych w przepływy, co wymaga skomplikowanego przetwarzania zebranych danych. Brak też zaawansowanych metadanych przepływu. W efekcie pozyskiwane tym sposobem dane wymagają dodatkowej obróbki, a w przypadku monitorowania wielu punktów także implementacji mechanizmu ich zbierania.
- **Monitorowanie bocznikowe** opiera się na kopiowaniu przez urządzenie sieciowe całości ruchu przekazywanego przez łącze na dodatkowy interfejs, na którym prowadzony jest nasłuch. Możliwości są praktycznie takie same, jak przy monitorowaniu systemowym, choć możliwe jest też użycie systemu IDS. Zaletą tego rozwiązania jest możliwość zebrania ruchu z wielu miejsc w jednym punkcie oraz uniknięcie wprowadzania wykrywalnych opóźnień w przekazywanym ruchu.
- **Monitorowanie na poziomie honeypota**, jak sama nazwa sugeruje, może być stosowane tylko w przypadku zbierania ruchu przez honeypoty. Niektóre rozwiązania honeypotowe zawierają wbudowaną funkcjonalność zbierania przepływów, która często bywa dość rozbudowana. Możliwe jest wiązanie pakietów w pełne przepływy, a nawet powiązanie ruchu ze zdarzeniami zarejestrowanymi na honeypocie, co umożliwia z dużą pewnością wskazanie przepływów rzeczywiście złośliwych.

- **Zbieranie ruchu na poziomie IDS** wykorzystuje możliwości rejestracji ruchu dostępne w rozwiązaniach IDS (np. snort). Zwykle zapewnione jest łączenie w całość wielopakietowych przepływów, a niezaprzeczalną zaletą rozwiązania jest to, że zbierane przepływy mogą od razu być dopasowywane do istniejących już, dowolnie skomplikowanych reguł. Daje to możliwość odmiennego traktowania przepływów, które są już rozpoznawane przez znane sygnatury i nie wymagają ponownej analizy.

Najwygodniejszym z praktycznego punktu widzenia, choć jednocześnie najtrudniejszym sposobem rejestracji ruchu jest wykorzystanie systemu IDS/IPS. W zastosowaniach badawczych zdecydowanie dominującym rozwiązaniem tego typu jest Snort [12]. Jest to narzędzie open source, dalece rozszerzalne i elastyczne, którego język opisu reguł (sygnatur) jest standardem de facto w branży – nawet te rozwiązania komercyjne, które wykorzystują własny język reguł, zwykle wspierają także format snort, lub przynajmniej zapewniają narzędzia do importu i eksportu reguł. Dzięki możliwości stosowania wtyczek możliwa jest bardziej złożona analiza pakietów i przepływów, niż pozwala na to język reguł. Od niedawna jednak wybór otwartego, darmowego rozwiązania do celów badawczych nie jest w pełni oczywisty. Suricata [13], obecnie dostępna już w wersji stabilnej, jak również dynamicznie rozwijana (najnowsza wersja osiągnęła status *release candidate*, podczas gdy wersja stabilna jest zaledwie pół roku starsza), jest kompatybilna ze Snortem – akceptuje reguły pisane dla Snort 2.9, choć autorzy zachęcają do tworzenia reguł optymalizowanych pod Suricatę. Podobnie jak Snort, Suricata jest systemem modularnym, o otwartych źródłach, łatwym do dostosowania. Znacznie bardziej zaawansowana wielowątkowość pozwala liczyć na większą wydajność na nowoczesnym sprzęcie i zdolność do wykorzystywania bardziej złożonych analiz. Dzięki zaawansowanej bibliotece HTP, odpowiadającej za normalizację i parsowanie protokołu HTTP, Suricata szczególnie dobrze radzi sobie z ruchem tego typu (WWW, serwisy internetowe, itp).

W chwili obecnej trudno wskazać jednoznacznie lepsze rozwiązanie. Za Snortem przemawia jego dojrzałość – niedługo projekt będzie obchodził piętnaste urodziny – i doświadczenie twórców. Suricata jest projektem młodym i można się jeszcze spodziewać dość licznych problemów przy jej wykorzystaniu, jest to jednak rozwiązanie bardzo obiecującym i przyszłościowo bez wątpienia godne uwagi.

4. Przegląd istniejących metod wytwarzania sygnatur

Poniżej opisano znane systemy automatycznego wytwarzania sygnatur. Ograniczono się przy tym do systemów, które zostały rzeczywiście zaimplementowane i przetestowane, a opracowania na ich temat są publicznie dostępne w literaturze naukowej.

Z uwagi na wymienione wcześniej cechy poszczególnych typów metod, za szczególnie interesujące uznano systemy syntaktyczne tekstowe wielotokenowe. Tylko systemy z tej grupy zostały opisane wyczerpująco. Pozostałe przedstawiono pokrótce w charakterze tła obrazującego całościowo tematykę tworzenia sygnatur.

Wśród istniejących systemów sygnaturowych można wyróżnić trzy wyraźnie odmienne grupy.

- **Systemy semantyczne**, przedstawione na przykładzie narzędzia Nemean. Istnieje niewiele działających, dobrze udokumentowanych w literaturze, otwartych rozwiązań z tej grupy – w zasadzie poza systemem Nemean, należącym do grupy algorytmów parsujących można jedynie wymienić pewne prace wykonywane w Europie, m. in. w ramach projektu WOMBAT, koncentrujące się na sygnaturach behawioralnych.
- **Systemy syntaktyczne pierwszej generacji**, tworzone do 2004/2005 roku, reprezentowane przez EarlyBird, Honeycomb, ARAKIS i Autograph. Są to systemy bardzo inżynierskie w swoich założeniach, pozbawione ambicji konstrukcji jak najdokładniejszej sygnatury – autorzy zadowalali się uzyskaniem użytecznych wyników dla rzeczywiście spotykanych w tamtym okresie zagrożeń. Algorytmy opierają się na wyszukiwaniu stosunkowo prostych zależności (podobieństw) między przepływami i można je sklasyfikować jako jednotokenowe lub mapujące (przy czym mapowanie jest tu czysto heurystyczne, niedokładne). Pewnym wyjątkiem jest system ARAKIS, który tworzy sygnatury wielotokenowe, ale używa do tego celu algorytmów jednotokenowych i klasteryzacji (patrz opis systemu).

  Rozwiązania z tej grupy są w zasadzie wystarczająco skuteczne w opisywaniu sygnaturami zagrożeń monomorficznych i oligomorficznych. Jako że zagrożenia tego typu do dziś dominują wśród zagrożeń aktywnych, systemy te nie straciły całkowicie praktycznej użyteczności. Gruntowne analizy skuteczności przeciwko zagrożeniom polimorficznym są jednak rzadkie w literaturze z tego okresu i w dużej mierze ograniczają się do ogólnych przemyśleń na bazie nieco dyskusyjnych założeń.
- **Systemy syntaktyczne drugiej generacji**, tworzone od 2005 roku. Ich wczesnym przedstawicielem, na pograniczu generacji, jest Polygraph, za w pełni dojrzałe rozwiązania tej generacji można natomiast uznać systemy Hamsa i Nebula. Starają się one wytworzyć sygnaturę wysokiej jakości, za zadanie do rozwiązania przyjmując opis zagrożeń polimorficznych behawioralnie, a więc wyjątkowo trudnych w analizie. Zagrożenia monomorficzne są w zasadzie ignorowane – z punktu widzenia systemu poprawnie wykrywającemu prawidłowości w robakach polimorficznych, monomorficzne można swobodnie traktować jako szczególny, trywialnie prosty przypadek. W tej generacji standardem jest rzetelna i głęboka matematyczna analiza problemu, co zapewnia jasne

sformułowanie założeń i uzasadnienie przewidywanej skuteczności metody. Sygnatury generowane przez takie nowoczesne systemy mogą być różnej postaci, jednak nieodmiennie są wielotokenowe, co nie dziwi w świetle wcześniejszych uwag na temat charakteru zagrożeń polimorficznych.

4.1. Systemy semantyczne na przykładzie systemu Nemean

Nemean [14] jest systemem semantycznym parsującym, prezentowanym jako przykład mechanizmu semantycznego. Systemy z tej rodziny są w zasadzie poza głównym nurtem badań przedstawionych w raporcie, jednak ich całkowite pominięcie mogłoby stworzyć mylne wrażenie, że aktywne badania prowadzone są jedynie nad mechanizmami syntaktycznymi. Prace nad algorytmami semantycznymi, w ostatnich latach zwłaszcza behawioralnymi, są bardzo intensywne. Jako przykład wybrano system parsujący, jako bardziej zbliżony do syntaktycznych i łatwiejszy do zwięzłego przedstawienia.

Choć mechanizm tworzenia sygnatury jest zupełnie inny, rezultatem analizy semantycznej są zwykłe sygnatury tekstowe. W przypadku Nemeana są to teoretycznie ścisłe wariantowe sygnatury wielotokenowe, choć najczęściej w poszczególnych wariantach występują tylko pojedyncze tokeny.

Schemat algorytmu jest prosty, aczkolwiek poszczególne czynności składowe są dość skomplikowane i ich szczegółowy opis wykraczałby daleko poza tematykę raportu. Przepływy wejściowe są normalizowane i zapisywane w ustrukturalizowanej postaci. Po ponownej normalizacji, tym razem wykorzystującej informacje o działaniu danej usługi, przepływy są grupowane w klastry, z których ostatecznie wydobywany jest wspólny dla wszystkich elementów automat skończony. Automat ten przetwarzany jest na sygnaturę poziomu połączeń. Następnie dane są ponownie grupowane w klastry obejmujące całe sesje i operacja jest powtarzana na poziomie sesji.

System taki ma wiele niezaprzeczalnych zalet. Jak argumentują autorzy, tworzenie sygnatur ze znajomością semantyki protokołu pozwala skutecznie wybrać nawet drobne stałe fragmenty, uwzględnić ataki wieloetapowe, zróżnicować znaczenie poszczególnych części sygnatury, tworzyć ogólne sygnatury z bardzo małej liczby próbek, a w końcu zawrzeć w sygnaturze użyteczne dla czytelnika informacje o sposobie działania zagrożenia.

Wadą podejścia jest jednak wymaganie bogatej wiedzy o szczegółach działania poszczególnych protokołów, która wykorzystywana jest na etapie normalizacji danych. System semantyczny nie jest zatem rozwiązaniem ogólnego przeznaczenia, lecz specjalizowanym do obserwacji pewnych popularnych protokołów. Niezbędna jest też jego aktualizacja w miarę zmian w wykorzystywanych złośliwie protokołach. Autorzy systemu traktują to jak drobny problem, jednak sami stworzyli system obsługujący jedynie ruch HTTP i NetBIOS, zapewniając w ten sposób dość dobre pokrycie znanych zagrożeń, ale pozostając daleko od uniwersalności.

Istnieje też możliwość konstrukcji innych algorytmów semantycznych nie wymagającej tak daleko idącej wiedzy o działaniu aplikacji, jaką muszą posiadać algorytmy behawioralne. Np.~w~cytowanym przez autorów artykule [15] rozważane jest wykorzystanie semantyki instrukcji procesorów rodziny x86.

4.2. Systemy syntaktyczne pierwszej generacji

Systemy opisane w poniższym rozdziale opisane są jedynie pokrótce, z uwagi na ich wartość historyczną jako ilustracja kierunku rozwoju metod wytwarzania sygnatur. Więcej uwagi poświęcono jedynie systemowi ARAKIS, z uwagi na dwie jego cechy:

- jest to system wdrożony do faktycznego użycia komercyjnego i użytkowany do dziś,
- choć używane algorytmy są zdecydowanie jednotokenowe, to końcowa sygnatura jest wielotokenowa, system można więc zaliczyć do grupy stanowiącej główny temat raportu.

4.2.1. EarlyBird

EarlyBird [16] jest pierwszym dobrze udokumentowanym w literaturze i faktycznie zaimplementowanym systemem podejmującym problem automatycznego tworzenia wzorców umożliwiających szybką klasyfikację ruchu na podejrzany i normalny. Jest to więc najstarszy system sygnaturowy. Należy do grupy mechanizmów syntaktycznych mapujących wytwarzających sygnatury rozmyte. Wykorzystuje proste mapowanie oparte na skrótach Rabina.

Skrót Rabina [4] to wartość liczbowa wyliczana z ciągu bajtów o ustalonej długości. Właściwości tego skrótu są dość interesujące:

- determinizm – taki sam podciąg ma zawsze ten sam skrót, bez względu na poprzedzające go wartości,
- prostota – skrót jest prostym wielomianem, jego wyliczanie jest więc bardzo mało kosztowne obliczeniowo i bardzo wydajne,
- potokowość – wartości skrótów kolejnych podciągów można wyliczać działając w trybie okna przesuwnego, przez przekształcenie skrótu podciągu poprzedniego, co dalej zwiększa wydajność obliczeniową i minimalizuje wprowadzane przez analizator opóźnienia.

W systemie EarlyBird dla każdego przepływu liczone są wszystkie możliwe skróty dla podciągów określonej długości. Nowe zagrożenie może zostać wykryte poprzez zauważenie w zbiorze podejrzanym odpowiednio dużej grupy powtarzających się skrótów. Mogą one wtedy stanowić prostą sygnaturę. Każdy przepływ, w którym wystąpi dostatecznie wiele skrótów charakterystycznych dla takiej grupy, można sklasyfikować jako wystąpienie odpowiedniego zagrożenia.

Sygnatury tego typu nie nadają się bezpośrednio do zastosowania w systemach IDS, wymagają odpowiedniej wtyczki. Jednocześnie można mieć wątpliwości co do selektywności tak tworzonych sygnatur. Dla zagrożeń polimorficznych rozwiązanie jest nieskuteczne – skróty części polimorficznej zdominują opis przepływu, a brak korelacji między długością nielicznych i krótkich fragmentów stałych a długością okna powoduje niewystarczające ich odwzorowanie.

4.2.2. Honeycomb

Honeycomb [17] jest drugim spośród najstarszych systemów automatycznego wytwarzania sygnatur. Zaprojektowano go do działania na przepływach rejestrowanych przez honeypoty. Jest to system wytwarzający ścisłe sygnatury jednotokenowe.

Po wstępnym grupowaniu według ogólnych charakterystyk, takich jak używany protokół, port docelowy, wspólne zakresy IP, itd, wytwarzanie sygnatur odbywa się osobno w każdej grupie poprzez wyszukanie najdłuższej wspólnej podsekwencji. Dane do wyszukiwania przygotowywane są na dwa różne sposoby:

- wyszukiwanie **poziome** odbywa się poprzez porównywanie tych samych w kolejności pakietów z różnych połączeń z tej samej grupy,
- wyszukiwanie **pionowe** polega na porównywaniu całych przepływów.

Tak uzyskane sygnatury są bardzo łatwe w użyciu i skuteczne dla zagrożeń monomorficznych. Wadą systemu jest natomiast bardzo duża ilość powstających sygnatur, która uniemożliwia automatyczne ich wykorzystanie w systemach IDS/IPS. W celu ograniczenia rozmiaru zbioru sygnatur stosowana jest jedynie prosta i niewystarczająco skuteczna agregacja sygnatur, eliminująca ze zbioru proste duplikaty i sygnatury zawierające się w innych.

4.2.3. Autograph

Autograph [3] to prostym system wytwarzający ścisłe sygnatury jednotokenowe. Jego najciekawszą cechą jest oryginalny algorytm ekstrakcji tokenów, opisany już w rozdziale 3.1.

System uwzględnia możliwość zaszumienia zbioru podejrzanego lub występowania w nim więcej niż jednego zagrożenia, wytwarzana jest więc nie jedna, a cały zbiór sygnatur. Wykorzystywany jest prosty algorytm zachłanny – po odrzuceniu tokenów występujących wyłącznie w przepływach pochodzących z jednego adresu IP (co jest mało prawdopodobne dla epidemii robaka) iteracyjnie wybierany jest token występujący w największej liczbie przepływów, dla których jeszcze nie wybrano charakterystycznego tokena. Procedura kończy się, gdy dostatecznie mało przepływów pozostaje bez przypisanego tokena, lub gdy najczęściej występujący token pokrywa zbyt małą część pozostałego zbioru przepływów. Pozostałe po zatrzymaniu procedury przepływy są odrzucane jako szum. Dla zbioru idealnego (tylko jedno zagrożenie bez szumu) podejście takie daje tylko jedną sygnaturę, w innych przypadkach powstanie ich zbiór pokrywający możliwie dużą część przepływów.

Choć autorzy algorytmu argumentują, że może on być skuteczny także przeciwko zagrożeniom polimorficznym, przedstawiane argumenty wydają się niedostateczne. Zagrożenia te mogą nie zawierać długich fragmentów stałych, a ustawienie krótkiej oczekiwanej długości tokena, choć może (ale nie musi) pozwolić na wykrycie rzeczywistej stałej części, może skutkować wieloma fałszywymi alarmami.

4.2.4. ARAKIS

ARAKIS to system wczesnego ostrzegania opracowany w Naukowej i Akademickiej Sieci Komputerowej. Zawiera on w szczególności moduł automatycznego wytwarzania sygnatur [18], który w opinii autora raportu jest najbardziej zaawansowanym i skutecznym w praktyce mechanizmem pierwszej generacji. Należy podkreślić, że niniejszy raport uwzględnia jedynie mechanizm stosowany w wersjach 1.x systemu – rozwijana obecnie wersja 2.0 ma zostać wyposażona w nowy, wielotokenowy mechanizm drugiej generacji, którego szczegóły nie są jednak obecnie znane.

Procedura generacji sygnatur jest w dużym stopniu połączeniem idei systemów EarlyBird i Honeycomb oraz automatycznej klasteryzacji. Pierwszą fazą analizy przepływów w systemie jest ich filtrowanie przy użyciu skrótów Rabina. Mechanizm jest podobny do EarlyBird, ale skróty nie są traktowane jak sygnatury – na ich podstawie wyodrębnia się jedynie grupy nietypowych, ale podobnych do siebie przepływów.

Dla każdej takiej grupy, nazywanej zdarzeniem, wyszukiwane są najdłuższe wspólne podciągi, określane jako sygnatury LCS. Tak samo jak w Honeycomb, wytwarzane są zarówno sygnatury poziome, jak i pionowe.

Zebrane sygnatury okresowo poddawane są klasteryzacji z zastosowaniem nieznacznie zmodyfikowanego algorytmu DBSCAN (patrz rozdział 3.2). Sygnatury LCS wchodzące w skład każdego z utworzonych klastrów są scalane w tak zwaną supersygnaturę poprzez zastępowanie różniących się fragmentów „dziurami”. Supersygnatura w terminologii ARAKIS jest zatem zwykłą ścisłą sygnaturą wielotokenową w formie ciągu tokenów, nadającą się do bezpośredniego użycia w systemach IDS/IPS (z zastrzeżeniem, że brak jest mechanizmu automatycznej weryfikacji specyficzności sygnatury z użyciem próbek ruchu niezłośliwego).

Należy podkreślić, że pierwsze kroki wielofazowego algorytmu, w których wykrywane są rzeczywiste podobieństwa, są realizowane przy użyciu algorytmów zdecydowanie jednotokenowych. Wprawdzie ostateczna sygnatura ma postać wielotokenową, jednak nie można oczekiwać od niej odporności na polimorfizm takiej, jaką dają algorytmy z założenia wielotokenowe. Stałe, wspólne dla wszystkich przepływów w klastrze i charakterystyczne dla danego zagrożenia fragmenty treści mogą zostać pominięte, jeśli nie mieściły się w najdłuższych wspólnych podciągach odpowiednich przepływów, czyli nie występowały w początkowych sygnaturach jednotokenowych. W porównaniu do mechanizmów jednotokenowych uzyskiwane sygnatury są więc znacznie ogólniejsze – nieistotne, przypadkowe fragmenty są skutecznie eliminowane – ale ograniczone do tych samych fragmentów przepływu.

4.3. Systemy syntaktyczne drugiej generacji

Niniejszy rozdział przedstawia drugą generację syntaktycznych mechanizmów wytwarzania sygnaturm opierającą się na algorytmach wielotokenowych.

4.3.1. Polygraph

Polygraph [5] systemem szczególnym – w przeciwieństwie do pozostałych prezentowanych rozwiązań stosuje kilka różnych metod wytwarzania sygnatur. Sygnatury te poddawane są ocenie z wykorzystaniem zbioru normalnego, dzięki czemu najgorsze z nich mogą zostać odrzucone. W efekcie system tworzy sygnatury zróżnicowane, co utrudnia ich zastosowanie. Są to jednak sygnatury dobrej jakości, o niskiej stopie błędów, zarówno fałszywych detekcji, jak i pominięć.

Pierwszym etapem tworzenia sygnatur, wspólnym dla wszystkich metod, jest wyszukanie tokenów. Jak wspomniano w rozdziale 3.1, Polygraph wyszukuje wszystkie tokeny wspólne dla odpowiednio wielu przepływów ze zbioru, dysponuje więc odpowiednimi fragmentami sygnatur, których stworzenie wymaga tylko odpowiedniego wyboru elementów.

Opis poszczególnych rodzajów sygnatur tworzonych przez Polygraph zostanie przedstawiony w formie osobnych podrozdziałów, gdyż w zasadzie mogą one być traktowane jak odrębne systemy.

4.3.1.1. Sygnatury złączeniowe

Sygnatury złączeniowe (ang. *conjunction signatures*) należą do typu ścisłych sygnatur wielotokenowych w formie zbioru tokenów. Powstają one w trywialnie prosty sposób – poprzez wybranie ze zbioru tokenów wszystkich tych, które występują we wszystkich przepływach podejrzanych.

Tak daleko idące usztywnienie zasad wyboru można uznać za nieco kontrowersyjne – w praktyce dopuszczenie tokenów, które nie występują tylko w niewielkiej liczbie przepływów podejrzanych, mogłoby zwiększyć tolerancję na zaszumienie zbioru podejrzanego.

Ograniczenie systemu Polygraph do tej jednej metody generacji pozwoliłoby na znaczne uproszczenie, tę samą funkcjonalność można bowiem uzyskać przez prostą zmianę parametrów samego procesu ekstrakcji tokenów, bez dalszego przetwarzania.

Procedura tworzenia sygnatur jest w tym przypadku bardzo wrażliwa na jakość klasyfikatora przepływów (czyli szum), a także na jednoczesne występowanie więcej niż jednego robaka. Wrażliwość tę zmniejsza wykorzystanie mechanizmu klasteryzacji hierarchicznej (patrz rozdział 3.2).

4.3.1.2. Sygnatury ciągowe

Sygnatury ciągowe (ang. *token sub-sequence signatures*) to najlepiej sprawdzający się w testach typ sygnatur systemu Polygraph. Są to ścisłe sygnatury wielotokenowe w formie ciągu tokenów. Wytwarzanie sygnatur bazuje na algorytmie Smitha-Watermana [2, 19] wyszukiwania optymalnego dopasowania (ang. *alignment*) ciągów znaków. Słabością metody jest stosowanie tego algorytmu jedynie do dwóch ciągów jednocześnie, przez co możliwe jest wybranie dopasowań uwzględniających podobieństwo niektórych przepływów znaczące dla danej pary, ale nietypowe dla całego zbioru. Zmniejszenie tego ryzyka następuje przez wykorzystanie wyników fazy ekstrakcji tokenów.

Tworzenie sygnatur wymaga poszerzenia alfabetu o dodatkowy znak, oznaczany γ, który oznacza dowolną liczbę dowolnych znaków z pierwotnego alfabetu. Przepływy w zbiorze podejrzanym zastępowane są ciągami należących do nich wyekstrahowanych w pierwszej fazie tokenów, połączonych znakami γ. Pozostawione zostają zatem jedynie te fragmenty, które są wspólne dla większych grup przepływów.

Algorytm składa się z następujących kroków:

1. Przetwórz wszystkie przepływy ze zbioru podejrzanego, pozostawiając w nich jedynie wydobyte tokeny, połączone znakami γ.
2. Pobierz dowolny przetworzony przepływ ze zbioru podejrzanego jako próbkę *A*.
3. Pobierz dowolny niewykorzystany jeszcze przetworzony przepływ ze zbioru podejrzanego jako próbkę *B*.
4. Stosując algorytm Smitha-Watermana z parametrami promującymi dopasowania zachowujące dłuższe ciągłe fragmenty tekstu dopasuj obie próbki.
5. Wytwórz nową próbkę na bazie uzyskanego dopasowania, zastępując różniące się fragmenty znakiem γ, a fragmenty tożsame pozostawiając bez zmian. Ewentualne sąsiadujące znaki γ powinny być łączone w jeden. Nowa próbka staje się próbką *A*.
6. Jeśli w zbiorze podejrzanym są jeszcze niewykorzystane przepływy, przejdź do punktu 3. W przeciwnym przypadku próbka *A* jest poszukiwaną sygnaturą, w której kolejne tokeny ciągu rozdzielone są znakiem γ.

Procedura jest zatem stosunkowo prosta. Co istotne, powstająca sygnatura nie musi składać się z tokenów, które zostały wyekstrahowane w fazie przygotowawczej, mogą występować w niej ich fragmenty.

Stosowanie klasteryzacji ma w tym przypadku jeszcze lepsze uzasadnienie – nawet niewielka liczba przepływów znacznie różniących się od pozostałych może wymusić usunięcie znacznego fragmentu skądinąd dobrej sygnatury. Trzeba jednak podkreślić, że kwadratowa względem liczby przepływów złożoność procedury klasteryzacji w połączeniu z również kwadratową, ale w funkcji długości przepływu, złożonością dopasowania powoduje, że tworzenie sygnatur tego typu w zaproponowany sposób jest dość kosztowne obliczeniowo.

4.3.1.3. Sygnatury bayesowskie

Ostatnim typem sygnatur tworzonych przez Polygraph są sygnatury bayesowskie. Są one interesujące o tyle, że dzięki nim Polygraph jest jedynym wymienionym w literaturze faktycznie zaimplementowanym systemem używającym metody z zakresu sztucznej inteligencji czy uczenia maszynowego (choć faktycznie najprostszej możliwej). System generuje rozmyte sygnatury wielotokenowe w formie zbiorów tokenów.

Jak sama nazwa wskazuje, algorytm jest prostą aplikacją naiwnego klasyfikatora Bayesa do zadania klasyfikacji przepływów. Poszczególnym tokenom przypisywane są wagi zależnie od prawdopodobieństwa ich wystąpienia w zbiorze podejrzanym i normalnym. Sygnaturę tworzy zbiór tokenów najlepiej różnicujących, a jej dopasowanie sprowadza się do próby wyszukania każdego z

tokenów w przepływie i obliczenia prawdopodobieństwa, że jest to przepływ złośliwy zależnie od tego, które tokeny w nim wystąpiły. Algorytm wymaga oczywiście ustawienia wartości progowych, zarówno na etapie wyboru tokenów tworzących sygnaturę, jak i samego dopasowania.

Wyniki w testach przeprowadzonych przez autorów Polygraphu są porównywalne z optymalnymi sygnaturami jednotokenowymi. Konkretne wyniki zależą od testu, jednak ogólnie sygnatury bayesowskie wydają się być równie dobre, jak złączeniowe, ale gorsze od ciągowych. Trzeba jednak podkreślić rolę właściwego ustawienia wartości progowych

4.3.2. Hamsa

Hamsa [6] tworzy ścisłe sygnatury wielotokenowe w formie multizbiorów tokenów. System jest z założenia stworzony przede wszystkim w celu wykrywania zagrożeń polimorficznych, w których atakujący ma pełną kontrolę nad treścią przepływu, z wyjątkiem fragmentów, które muszą być niezmienne dla udanego ataku. Zakłada się zatem, że tokeny charakteryzujące zagrożenie mogą być liczne, ale krótkie, przez co pojedynczo są bezużyteczne z uwagi na liczne fałszywe alarmy. Celem jest wytworzenie sygnatury, która pokrywa jak największą część zbioru podejrzanego bez przekroczenia założonego progu fałszywych alarmów. Mechanizm opiera się na ciekawym modelu matematycznym, który zostanie poniżej w zarysie przedstawiony.

Wszystkie tokeny wyekstrahowane ze zbioru podejrzanego (jak opisano w rozdziale 3.1 jest to realizowane przy użyciu tablic sufiksowych) są oceniane pod względem częstości występowania w zbiorze normalnym. Model ataku zakłada, że o ile występowanie tych tokenów jest skorelowane, kiedy celem jest atak, to występowanie ich w przepływach normalnych jest znacznie bardziej niezależne, a więc prawdopodobieństwo wystąpienia kilku z nich jednocześnie jest znacznie niższe, niż każdego z osobna. Oznaczmy współczynnik fałszywych alarmów, czyli odsetek przepływów zbioru normalnego, w których występują jednocześnie wszystkie tokeny z jakiegoś multizbioru X przez FP_X (jeśli jakiś token występuje w multizbiorze kilkakrotnie, to tyle razy musi się pojawić w różnych miejscach przepływu).

Atakujący może wygenerować dowolną treść przepływu, ale musi ona zawierać pewien określony zbiór tokenów I (ponieważ nie muszą one być różne, jest to multizbiór). Po porównaniu ze zbiorem normalnym, wprowadzany jest w zbiorze I porządek tokenów, tzn. $I=\{\hat{t_1}, \hat{t_2}, \ldots \hat{t_k}\}$, gdzie:

$$\begin{aligned} FP_{\hat{t_1}} &\le FP_{t_i} && \forall i \\ FP_{\hat{t_1},\hat{t_2}} &\le FP_{\hat{t_1},t_i} && \forall i > 1 \\ FP_{\hat{t_1},\ldots\hat{t_j}} &\le FP_{\hat{t_1},\ldots\hat{t_{j-1}},t_i} && \forall j \forall i > j-1, \end{aligned}$$

czyli pierwszy token to ten, który występuje najrzadziej wśród przepływów normalnych, drugi to ten, który występuje w nich najrzadziej łącznie z pierwszym, itd. Zwróćmy uwagę, że poza dostępnością zbiorów podejrzanego i normalnego oraz początkowej listy tokenów, nie było dotąd konieczne przyjęcie żadnych szczególnych założeń. Uporządkowanie takie jest zawsze możliwe do przeprowadzenia. Przypadki równych częstości niektórych kombinacji są dopuszczalne i nie określono jasno sposobu rozwiązywania takich sytuacji, ale można wtedy przyjąć dowolny wybór bez znaczenia dla skuteczności algorytmu.

Istotne założenia pojawiają się dopiero w tzw. modelu napastnika (ang. *adversary model*). Model napastnika $\Gamma(k^*, u(1), \ldots u(k^*))$ zakładany w systemie Hamsa narzuca ograniczenie na prawdopodobieństwa fałszywych alarmów w zbiorze *I*, a ściślej w jego pierwszych k^* tokenach. Prawdopodobieństwo a priori jednoczesnego wystąpienia *k* pierwszych tokenów w przepływie normalnym, szacowane jako prawdopodobieństwo a posteriori wyznaczone na posiadanym zbiorze normalnym, nie może być większe, niż *u(k)*.

Przy tym założeniu, dobra sygnatura może zostać wytworzona przez prosty algorytm zachłanny, który w kroku *k* wyszukuje wszystkie tokeny, których dołączenie do sygnatury nie powoduje przekroczenia ograniczenia, a następnie wybiera ten, który daje najlepsze pokrycie zbioru podejrzanego. Problemem jest jedynie kryterium zatrzymania. Wykonywanie zawsze k^* kroków nie jest dobrym rozwiązaniem, chociażby dlatego, że dane zagrożenie może mieć mniej stałych fragmentów, niż założono w modelu. Najprostszym warunkiem stopu jest osiągnięcie prawdopodobieństwa fałszywego alarmu mniejszego od założonego progu.

Analiza teoretyczna wykazuje wysoką skuteczność takiego algorytmu, jednak w praktyce jakość uzyskiwanych sygnatur bywa różna. Powodem jest zbytnie uproszczenie wyboru tokena, w którym kryteria są stosowane w porządku leksykograficznym: odsetek fałszywych alarmów jest traktowany jedynie jako kryterium wstępne, binarne (przekracza próg lub nie), natomiast wybór dokonywany jest już wyłącznie na bazie pokrycia zbioru. Sygnatura obejmująca o jeden przepływ więcej niż druga zostanie zatem uznana za lepszą, nawet jeśli daje odsetek fałszywych alarmów bliski progowi, a druga bliski zeru. Autorzy przyjmują proste, skalaryzacyjne rozwiązanie problemu wielokryterialności, wprowadzając funkcję oceny w formie $\text{score}(COV_S, FP_S, LEN_S) = -\log_{10}(\delta + FP_S) + a\,COV_S + b\,LEN_S$, gdzie $a >> b$, a mała δ zabezpiecza przed dążeniem pierwszego składnika do nieskończoności dla bardzo selektywnych sygnatur. Tokeny są wybierane według wartości tej funkcji.

Parametry mogą być różne, autorzy twierdzą, że $\delta = 10^{-6}$, $a = 20$ i $b=0.01$ dają dobre rezultaty. Taka funkcja promuje początkowo sygnatury o dużym pokryciu, ale dla bardzo niskich prawdopodobieństw fałszywych alarmów wybiera sygnatury precyzyjniejsze. Dodatkowo lekko preferowane są sygnatury krótkie, przez co token nieznacznie poprawiający parametry sygnatury nie zostanie do niej dołączony.

Uzyskane pokrycie może być różne, algorytm nie zawiera jego ograniczenia. Algorytm powinien być stosowany iteracyjnie – po wytworzeniu pierwszej sygnatury należy usunąć ze zbioru pasujące przepływy i powtórzyć całą operację, aż do pokrycia wymaganej części zbioru podejrzanego, zależnie od tego, jak dużą część zbioru jesteśmy gotowi uznać za szum. W ten sposób Hamsa zostaje uodporniona na występowanie wielu różnych zagrożeń jednocześnie.

Sygnatury uzyskiwane przez system Hamsa są bardzo dobrej jakości. Nie byłoby uprawnione, bez względu na subiektywną opinię, bezpośrednie porównanie wyników z najbliższym konkurentem, czyli systemem Polygraph, bez pewności, że w prezentowanych w artykułach testach wykorzystano te same próbki, a w zasadzie nawet z pewnością, że tak nie jest. Dla poszczególnych robaków użytych w testach Hamsa uzyskuje jednak bardzo dobre rezultaty. Jest przy tym, dzięki lepszej złożoności, co najmniej kilkudziesięciokrotnie szybsza, niż Polygraph, co z punktu widzenia

praktycznej użyteczności można uznać za deklasację. Jako że wytworzone przez nią sygnatury są też bardzo łatwe do wykorzystania w praktyce, należy ją uznać za bardzo udany system.

Nie oznacza to bynajmniej, że nie były podejmowane próby jego udoskonalenia. Można chociażby wymienić system Lisabeth [20], próbujący udoskonalić sposób wyboru tokenów do sygnatury, zwiększając w ten sposób odporność na próby manipulacji systemem sygnaturowym, w szczególności nowoopracowany atak prowadzący do tworzenia sygnatur złej jakości.

4.3.3. Nebula

System Nebula [7] dąży bezpośrednio do uzyskania najdłuższej wspólnej podsekwencji podobnych podejrzanych przepływów, oczywiście w sposób heurystyczny (problem NP-trudny). Wytwarzana jest zatem ścisła sygnatura wielotokenowa w postaci ciągu tokenów pozycjonowanych.

Pierwszym krokiem przetwarzania w systemie Nebula jest wyodrębnienie grup przepływów, dla których tworzone będą sygnatury. W tym celu wykorzystywany jest algorytm klasteryzacji grafowej, opisany w rozdziale 3.2. Dalsze kroki wykonywane są oddzielnie dla każdego klastra, który składa się z odpowiedniej liczby przepływów. Pojedyncze przepływy i zbyt małe klastry traktowane są jako szum, sygnatury dla nich wytworzone zostaną dopiero, kiedy będą częściami większego klastra.

Pierwszym krokiem tworzenia sygnatury klastra jest wydobycie tokenów. Jak wspomniano w rozdziale 3.1, stosowany jest tu szybki algorytm o złożoności liniowej, oparty na tablicach sufiksowych, jako na oszczędnej implementacji uogólnionych drzew sufiksowych. Zaletą tego podejścia jest posiadanie informacji o położeniu tokenów w poszczególnych przepływach.

Dalsze tworzenie sygnatury opiera się na prostym algorytmie zachłannym. W pierwszym kroku wybierany jest najdłuższy wspólny token, dla którego ustalany jest przedział indeksów w ramach przepływu, na których może występować. W kolejnych krokach wybierany jest zawsze najdłuższy token, którego przedział możliwych indeksów nie przecina się z analogicznym przedziałem żadnego z tokenów już wchodzących w skład sygnatury. Ostatecznie otrzymywany jest ciąg tokenów, które muszą wystąpić w ściśle określonej kolejności, przy czym dla każdego z nich określone jest najwcześniejsze i najpóźniejsze możliwe położenie. Sygnatury takie dają się łatwo wyrazić w wielu językach reguł, między innymi snort.

System jest jak widać stosunkowo zaawansowany, choć prosty konstrukcyjnie. W publikacji dobrze przeanalizowano jego złożoność, umotywowano skuteczność, a przeprowadzone testy wskazują na dobrą jakość uzyskiwanych rezultatów.

5. Możliwości rozwojowe

Jednym z zagadnień, których badanie było przewidziane w ramach grantu, jest zwiększenie wydajności metod wytwarzania i dopasowywania sygnatur poprzez wykorzystanie mocy obliczeniowej kart graficznych. Po dokonaniu przeglądu literatury do badań wybrano systemu Hamsa i Nebula. Ponieważ prace trwają i zostaną udokumentowane w innym raporcie, szczegółowy opis zostanie tu pominięty.

Innym wątkiem badań są udoskonalenia i modyfikacje znanych metod. W tym przypadku warto wskazać najważniejszą wadę nowoczesnych metod wielotokenowych – trudności w wykorzystaniu w warunkach pracy ciągłej, szczególnie jeśli powstające sygnatury mają być wykorzystywane jako punkt wyjścia do ręcznej analizy zagrożeń. Algorytmy te do działania potrzebują odpowiednio dużego zbioru przepływów podejrzanych (i w niektórych przypadkach dodatkowo zbioru normalnego). Muszą zatem działać okresowo, uruchamiane tylko wtedy, kiedy pojawia się dostatecznie duża porcja nowych danych. Niestety, ponieważ – w przeciwieństwie np. do klastrowych supersygnatur systemu ARAKIS – rozwiązania te nie mają pamięci wcześniejszych uruchomień, możliwe jest, że dla tego samego robaka w kolejnych uruchomieniach powstaną dwie różne sygnatury (różniące się chociażby kolejnością tokenów). Oznacza to, że pojawienie się nowej, wcześniej nie widzianej sygnatury jest zjawiskiem normalnym i nie niesie ze sobą informacji o prawdopodobnym nowym zagrożeniu.

Interesujący jest też fakt, że nie ma wyraźnie dominującego formatu sygnatury wielotokenowej – zależnie od rozwiązania pojawiają się zarówno sygnatury o tokenach swobodnych, jak i pozycjonowanych, zarówno w formie zbiorów, ciągów, jak i multizbiorów tokenów. W każdym systemie stosowana jest postać taka, jaka jest najwygodniejsza dla podstawowego algorytmu. Brak zatem prób uzyskania sygnatur możliwie bogatych w informacje.

W związku z powyższymi problemami podjęte zostały prace nad modyfikacjami istniejących metod. Jako podstawowy system sygnaturowy przyjęto w nich system Hamsa, przy czym założono, że oryginalny algorytm nie będzie modyfikowany – zostaną natomiast wypróbowane możliwości postprocessingu wytworzonych sygnatur.

Pierwszym badanym udoskonaleniem jest dodanie do sygnatur metadanych o podobieństwie do starszych sygnatur. Nowa sygnatura jest porównywana ze starą (każda z każdą), jeśli są dostatecznie podobne, lub dają te same rezultaty na zbiorze testowym składającym się z części zbioru normalnego i odpowiednich fragmentów starego i nowego zbioru podejrzanego, to przyjmuje się, że nowa jest uaktualnieniem lub alternatywną formą starej. Z punktu widzenia systemu nie jest to zmiana istotna, natomiast z punktu widzenia specjalisty bezpieczeństwa obsługującego incydent jest ona bezcenna, pozwala bowiem traktować wystąpienia dopasowań obu sygnatur jako kontynuację jednego zjawiska.

Drugie ciekawe zagadnienie to udoskonalanie sygnatury bez dodawania następnych tokenów. Proponowany jest mechanizm, który dla każdej wygenerowanej sygnatury sprawdza wystąpienia jej tokenów w zbiorze podejrzanym i normalnym. Następnie ustalane są zależności międzytokenowe. W efekcie sygnatury systemu Hamsa mogą zostać wzbogacone o:

- wymuszoną kolejność niektórych tokenów,
- ograniczony zakres indeksów na których może występować dany token,
- ograniczony przedział możliwych odległości między dwoma tokenami.

Każda taka zmiana wprowadzana jest tylko i wyłącznie wtedy, kiedy podnosi selektywność sygnatury bez utraty jej czułości. Tego rodzaju udoskonalenia sygnatury mogą stać się kluczowe w przypadku wykrycia robaka rzeczywiście wykorzystującego polimorfizm behawioralny w bardzo skuteczny sposób. Jeśli liczba rzeczywistych stałych fragmentów przepływu będzie zbyt mała, to Hamsa może nie mieć możliwości utworzenia dostatecznie selektywnej sygnatury bez umieszczania w niej przypadkowych wielokrotnych wystąpień tego samego ciągu w części zmiennej, co prowadzi do zwiększenia liczby i pogorszenia jakości sygnatur. W tej sytuacji wytwarzanie sygnatury należałoby zatrzymać po wyczerpaniu tokenów o bardzo dużym pokryciu w zbiorze podejrzanym, a dalsze zwiększenie selektywności uzyskiwać właśnie wykorzystując informacje o położeniu tokenów w przepływie.

Oczywiście zastosowanie tego rodzaju postprocessingu zmieni postać sygnatury Hamsy – nie będzie to już multizbiór, ale graf pozycjonowanych tokenów. Jako że obecnie nie istnieją systemy tego typu, efektywne sposoby przekładania powstałego grafu na język reguł snort są również obecnie przedmiotem badań.

Bibliografia

[1] P. Li, M. Salour, X. Su, *A Survey of Internet Worm Detection and Containment*, IEEE Communications Surveys & Tutorials, 1st Quarter 2008, Vol. 10, No. 1, pp. 20-35.

[2] D. Gusfield, *Algorithms on strings, trees, and sequences: computer science and computational biology*, Cambridge University Press New York, NY, USA, 1997, ISBN:0-521-58519-8.

[3] H.-A. Kim, B. Karp. 2004. *Autograph: Toward Automated, Distributed Worm Signature Detection.* In Proceedings of the 13th conference on USENIX Security Symposium - Volume 13 (SSYM'04), Vol. 13. USENIX Association, Berkeley, CA, USA

[4] R.M. Karp, M.O. Rabin, *Efficient randomized pattern-matching algorithms*, IBM Journal of Research and Development - Mathematics and computing, vol. 31, no. 2, 1987, pp. 249-260.

[5] J. Newsome, B. Karp, D. Song, *Polygraph: Automatically Generating Signatures for Polymorphic Worms*, In: 2005 IEEE Symposium on Security and Privacy (S&P'05). (pp. 226 - 241). IEEE Computer Society: Los Alamitos, US.

[6] Z. Li , M. Sanghi , Y. Chen , M. Kao , B. Chavez, *Hamsa: Fast Signature Generation for Zero-Day Polymorphic Worms with Provable Attack Resilience*, In SP '06: Proceedings of the 2006 IEEE Symposium on Security and Privacy (S\&P'06).

[7] T. Werner, C. Fuchs, E. Gerhards-Padilla, P. Martini, *Nebula – Generating Syntactical Network Intrusion Signatures*, 4th International Conference on Malicious and Unwanted Software (MALWARE), pp 31-38, 2009.

[8] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *A density-based algorithm for discovering clusters in large spatial databases with noise*. In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96). AAAI Press. pp. 226?231. ISBN 1-57735-004-9, 1996.

[9] В.И. Левенштейн (1965). *Двоичные коды с исправлением выпадений, вставок и замещений симболов*. Доклады Академий Наук СССР 163 (4): 845–8. Appeared in English as: Levenshtein VI (1966). "Binary codes capable of correcting deletions, insertions, and reversals". Soviet Physics Doklady 10: 707–10.

[10] F.J. Damerau, *A technique for computer detection and correction of spelling errors*, Communications of the ACM, vol. 7, no. 3, 1964, pp.171-176.

[11] V. Roussev, G. G. Richard III , L. Marziale, *,Multiresolution Similarity Hashing*, Digital Investigation, Vol. 4, Supplement, September 2007, Pages 105-113.

[12] M. Roesch, *Snort – Lightweight Intrusion Detection for Networks.* In Proc. of the 13th Conference on Systems Administration, pp. 229–238, Seattle, WA, 1999.

[13] Strona domowa projektu Suricata, Open Information Security Foundation. http://www.openinfosecfoundation.org/

[14] V. Yegneswaran, J. T. Giffin, P. Barford, S. Jha, *An Architecture for Generating Semantics-Aware Signatures*, In: Proceedings of the 14th conference on USENIX Security Symposium (SSYM'05), Vol. 14. USENIX Association, Berkeley, CA, USA.

[15] M. Christodorescu, S. Seshia, S. Jha, D. Song, R.E. Bryant, *Semantics-Aware Malware Detection*, In IEEE Symposium on Security and Privacy, Oakland, California, May 2005.

[16] S. Singh, C. Estan, G. Varghese i S. Savage, *The EarlyBird System for Real-time Detection of Unknown Worms*, Technical Report CS2003-0761, CSE Department, UCSD, Aug. 2003.

[17] C. Kreibich, J. Crowcroft, *Honeycomb – Creating Intrusion Detection Signatures Using Honeypots*, 2nd Workshop on Hot Topics in Networks (HotNets-II), 2003, Boston, USA.

[18] P. Kijewski, *Metody automatycznego wytwarzania sygnatur zagrożeń sieciowych*, CERT Polska SECURE Conference, October 2005.

[19] T.F. Smith, M.S. Waterman, *Identification of Common Molecular Subsequence*. Journal of Molecular Biology vol. 147, 1981, pp. 195–197.

[20] L. Cavallaro, A. Lanzi, L. Mayer, M. Monga, *LISABETH: Automated Content-Based Signature Generator for Zero-day Polymorphic Worms.* In: Proceedings of the fourth international workshop on Software engineering for secure systems SESS'08, Leipzig, Germany, 2008, pp. 41-48.