GA-NIDS: System wykrywania włamań sieciowych oparty na algorytmie genetycznym

Anup Goyal
Inżynieria elektryczna i informatyka Ford 2-207, Northwestern University Evanston, Illinois
anup-goyal@northwestern.edu

Chetan Kumar
Electrical Engineering and Computer Science Tech L-460, Northwestern University Evanston, Illinois
chetan-k@northwestern.edu

ABSTRAKT

Wykrywanie złośliwych połączeń w sieciach komputerowych jest rosnącym problemem, motywującym do szeroko zakrojonych badań w informatyce w celu opracowania lepszych systemów wykrywania włamań (IDS). W tym artykule przedstawiamy podejście oparte na uczeniu maszynowym, znane jako algorytm genetyczny (GA), do identyfikacji takich szkodliwych/atakujących połączeń. Algorytm bierze pod uwagę różne cechy połączeń sieciowych, takie jak typ protokołu, usługa sieciowa w miejscu docelowym i status połączenia, aby wygenerować zestaw reguł klasyfikacji. Każda reguła w zestawie reguł identyfikuje określony typ ataku. W tym eksperymencie zaimplementowaliśmy GA i wytrenowaliśmy go na zestawie danych KDD Cup 99, aby wygenerować zestaw reguł, który można zastosować do IDS w celu identyfikacji i klasyfikacji różnych typów połączeń atakujących. Dzięki naszemu eksperymentowi opracowaliśmy zestaw reguł składający się z sześciu reguł do klasyfikacji sześciu różnych typów połączeń atakujących, które dzielą się na dwie klasy, a mianowicie ataki odmowy usługi i ataki sondujące. Wygenerowana reguła działa ze 100% dokładnością przy wykrywaniu połączeń typu odmowa usługi i ze znaczną dokładnością przy wykrywaniu połączeń sondujących. Wyniki naszego eksperymentu dały obiecujące rezultaty w zakresie zastosowania algorytmu genetycznego do wykrywania włamań do sieci.

1. WPROWADZENIE

Wykrywanie włamań w celu identyfikacji ataków na systemy komputerowe od dłuższego czasu stanowi wyzwanie w dziedzinie bezpieczeństwa sieci. Oprogramowanie wykrywające włamania do sieci chroni sieć komputerową przed nieautoryzowanym użyciem, zapobiegając w ten sposób złośliwym działaniom. Zadaniem uczenia się wykrywacza włamań jest zbudowanie klasyfikatora (tj. modelu predykcyjnego) zdolnego do rozróżnienia między atakiem/włamaniem ("złymi" połączeniami), a normalnymi lub dobrymi połączeniami.

Biorąc pod uwagę rosnące problemy z bezpieczeństwem sieci i potrzebę opracowania zaawansowanych i solidnych rozwiązań, w 1999 roku zorganizowano konkurs KDD Cup, zapraszając naukowców z całego świata do zaprojektowania innowacyjnych metod budowy IDS na zbiorze danych treningowych i testowych, popularnie nazywanym zbiorem danych KDD Cup 99 [3]. Od tego czasu różne techniki uczenia maszynowego, takie jak klasyfikatory Bayesa i drzewa decyzyjne, zostały przeszkolone na zestawie danych KDD Cup 99, aby nauczyć się normalnych i niespójnych wzorców z danych testowych, a tym samym wygenerować klasyfikatory, które są w stanie wykryć atak włamania [1]. Problemem wykrywania włamań sieciowych jest nie tylko identyfikacja połączeń ataków, ale także wiedza, do jakiego typu ataku należy dane połączenie [3]. Ograniczeniem tych klasyfikatorów jest to, że generują one zunifikowaną regułę dla wszystkich typów ataków. W rezultacie, chociaż algorytmy te dobrze radzą sobie z segregowaniem ataków

od normalnych połączeń, ich wydajność nie jest tak wysoka, jeśli chodzi o identyfikację typu ataku, jakim jest/było połączenie. Można to intuicyjnie wytłumaczyć faktem, że pojedyncza reguła nie może dokładnie sklasyfikować wszystkich typów ataków. To zasadniczo tworzy problem i motywację stojącą za tym projektem. Chcielibyśmy mieć system wykrywania włamań, który jest w stanie przewidzieć rodzaj przychodzących ataków innych niż identyfikacja połączeń atakujących. Spośród różnych technik uczenia maszynowego wybraliśmy algorytm genetyczny jako dobry sposób na znalezienie skutecznego rozwiązania problemu.

Algorytmy genetyczne to kolejne podejście do uczenia maszynowego oparte na zasadach obliczeń ewolucyjnych [4]. Obejmują one koncepcję teorii Darwina i doboru naturalnego w celu wygenerowania zestawu reguł, które można zastosować na zbiorze testowym w celu sklasyfikowania włamań. Naukowcy badali wykorzystanie GA w wykrywaniu włamań i zgłaszali bardzo wysokie wskaźniki sukcesu, ale na zestawach danych innych niż KDD 99 Cup, takich jak zestaw danych DARPA [2][4][5]. Zaprojektowaliśmy GA, który generuje zestaw reguł, w którym każda reguła została wygenerowana ewolucyjnie w celu sklasyfikowania każdego typu ataku w zestawie danych treningowych i testowych. Uzasadnieniem wyboru GA do tego zadania jest nieodłączna cecha ewolucyjna algorytmu, która pozwala nam zdefiniować własną funkcję fitness, na podstawie której wybierane są tylko te elementy, reguły lub hipotezy, które spełniają nasze kryterium fitness (opisane później). Możemy zatem sprawić, że algorytm wybierze tylko te jednostki, które najlepiej klasyfikują typy ataków na każdym etapie jego ewolucji.

Mając na uwadze to podejście, zaprojektowaliśmy i wdrożyliśmy GA, koncentrując się na osiągnięciu wysokiego wskaźnika dokładności przewidywania >95% i minimalnego wskaźnika wyników fałszywie dodatnich na zbiorze danych KDD Cup 99. Ten wskaźnik dokładności predykcji stanowi dla nas punkt odniesienia do testowania wydajności GA. W tym artykule przedstawiamy szczegóły dotyczące projektu GA, konfiguracji eksperymentalnej i wyników naszej implementacji. Jak dotąd, z 24 różnych typów ataków w zestawie danych treningowych, byliśmy w stanie wygenerować zestaw reguł zawierający sześć różnych reguł należących do dwóch oddzielnych klas (wyjaśnionych w następnej sekcji), z których każda może poprawnie klasyfikować etykiety ataków, które zostały wybrane do poprawnej identyfikacji, a tym samym późniejszego przewidywania. Trening algorytmu został przeprowadzony na 10% zestawie danych treningowych KDD Cup (wyjaśnionym w następnej sekcji), podczas gdy testowanie zostało przeprowadzone na całym 100% oznakowanym zestawie. Połączenie reguł daje nam znaczną dokładność (do 98%) na zestawie danych treningowych i 92% dokładność na zestawie danych testowych, uzasadniając w ten sposób wybór, wydajność i zastosowanie algorytmów genetycznych do wykrywania włamań do sieci.

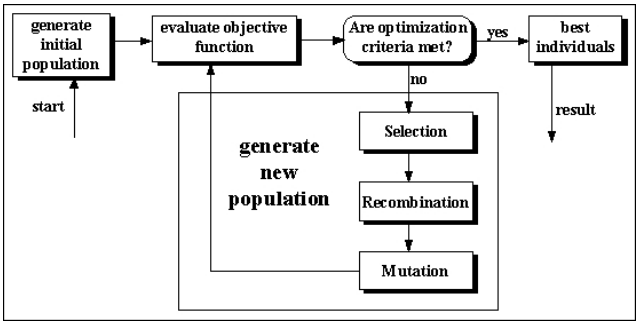
2. ZESTAW DANYCH TRENINGOWYCH I TESTOWYCH

Wykorzystaliśmy zestaw danych KDDCUP 99 do trenowania i testowania klasyfikatora systemu. Zestaw danych został dostarczony przez MIT Lincoln Labs. Zawiera on szeroką gamę włamań symulowanych w wojskowym środowisku sieciowym, skonfigurowanym w celu uzyskania dziewięciu tygodni surowych danych zrzutu TCP/IP dla sieci lokalnej (LAN) symulującej typową sieć LAN Sił Powietrznych USA. Sieć LAN działała tak, jakby była prawdziwym środowiskiem Sił Powietrznych, najeżonym wieloma atakami. Dlatego też jest to zestaw danych o wysokim poziomie zaufania i wysokiej jakości.

Stworzyli oni środowisko do zbierania zrzutów TCP/IP z hosta znajdującego się w symulowanej sieci wojskowej. Każde połączenie TCP/IP jest opisane przez 41 dyskretnych i ciągłych cech (np. czas trwania, typ protokołu, flaga itp.) i oznaczone jako normalne lub jako atak, z dokładnie jednym określonym typem ataku (np. Smurf, Perl itp.). Ataki dzielą się na cztery główne kategorie: (i) Ataki typu Denial of Service (DOS), w których atakujący przytłacza hosta ofiary ogromną liczbą żądań. (ii) Ataki typu User to Root (U2R), w których atakujący lub haker próbuje uzyskać prawa dostępu od normalnego hosta, aby na przykład uzyskać dostęp roota do systemu. (iii) Ataki zdalne na użytkownika (R2L), w których intruz próbuje wykorzystać luki w zabezpieczeniach systemu w celu kontrolowania zdalnej maszyny za pośrednictwem sieci jako użytkownik lokalny. (iv) Sondowanie, w którym atakujący próbuje zebrać przydatne informacje o maszynach i usługach dostępnych w sieci w celu poszukiwania exploitów. W naszym systemie wykorzystaliśmy 10% zestawu treningowego opublikowanego przez Lincoln Labs, który zawiera 494 021 połączeń. Nasz zestaw testowy to cały zestaw oznaczonych połączeń składający się z około 4,9 miliona połączeń. W ten sposób, korzystając z całego zestawu danych, byliśmy w stanie przetestować nasz system na niewidocznych połączeniach.

Na obecnym etapie implementacji udało nam się wygenerować zestaw reguł składający się z sześciu reguł, z których każda poprawnie klasyfikuje sześć różnych etykiet ataków. Wybraliśmy trzy najlepsze rozkłady etykiet ataków z dwóch klas ataków: DOS i Probe, w 10% zestawie danych treningowych. Te sześć etykiet ataków to: smurf, Neptune, land: typ ataków DOS oraz satan, ipsweep i portsweep: typ ataków Probe.

3. ARCHITEKTURA NID



Rysunek 1: Architektura NID dla naszego pierwszego etapu wdrożenia.

Rysunek 1 przedstawia architekturę IDS wyższego poziomu. W oparciu o nasze kryteria kondycji wybieramy najlepiej dopasowane osobniki z każdej populacji, które są w stanie wykrywać ataki i poprawnie klasyfikować je do odpowiednich etykiet podataków, tj. smerfów itp. Następnie stosujemy krzyżowanie i mutację

w celu wygenerowania nowych reguł, które są bardziej ukierunkowane na wykrywanie ataków. W ciągu serii pokoleń generujemy bazę reguł, która zawiera jedną regułę poprawnie klasyfikującą typ ataku, do którego została przeszkolona. Po zakończeniu tego projektu, biorąc pod uwagę, że w zestawie danych szkoleniowych znajdują się 24 etykiety ataków, nasza baza reguł będzie zawierać 24 różne reguły klasyfikujące każdy typ ataku. Do tego etapu wdrożenia, jak wspomniano wcześniej, nasza baza reguł poprawnie klasyfikuje sześć różnych typów ataków.

4. METODY

4.1 KODOWANIE POŁĄCZEŃ

Jak wspomniano wcześniej, każde połączenie sieciowe w zbiorze danych KDD Cup zawiera 41 pól/cech, z których cztery, w tym etykieta, są ciągami znaków, 14 pól zawiera wartości typu float w zakresie 0,00-1,00, podczas gdy pozostałe 23 pola zawierają wartości całkowite. Przy 41 różnych polach o różnych typach wartości, przestrzeń wyszukiwania hipotez, tj. liczba możliwych osobników w każdej populacji, staje się niezgłębiona, a zatem będzie wymagać bardzo dużego rozmiaru populacji, wymagającego obliczeń w ultraskali w celu wybrania odpowiednich osobników. Z tego powodu ograniczyliśmy naszą przestrzeń hipotez, obliczając górną i dolną granicę dla każdego pola ze zbioru treningowego i zezwoliliśmy na wartości tylko w tym zakresie w losowej populacji, którą generujemy na początku eksperymentu. Zrobiliśmy to za pomocą poleceń konsoli linux, aby znaleźć maksymalne i minimalne wartości dla każdego pola. Pola niosące wartości typu string zostały przedstawione za pomocą liczb dodatnich zaczynających się od 0, przy czym każdemu nowemu typowi ciągu przypisano zwiększoną liczbę. Na przykład, dla pola "protocol", icmp = 0, udp = 1. Inne pola niosące wartości typu float zostały włączone do genotypu, tj. zakodowanego ciągu znaków, tak jak pojawiają się w połączeniu. Zgodnie z zasadami GA, na początku eksperymentu tworzymy początkową populację, która zawiera osobniki, których geny są wybierane losowo z zakresu, w którym mogą się różnić. Każda populacja składa się ze 100 osobników.

4.2 FUNKCJA FITNESS

Po wygenerowaniu populacji początkowej, użyjemy funkcji fitness jako metryki do wyboru pasujących osobników, które zostaną poddane krzyżowaniu i mutacji w celu utworzenia populacji następnej generacji. Nasza funkcja fitness jest określona wzorem: **F = a/A - b/B,** gdzie a reprezentuje liczbę połączeń ataku, które osobnik poprawnie klasyfikuje z całkowitej liczby A ataków, b reprezentuje liczbę normalnych połączeń, które sieć poprawnie klasyfikuje z całkowitej liczby B normalnych połączeń w populacji. Stąd widzimy, że nasza wartość funkcji fitness będzie leżeć w obszarze [-1,1]. Wartość dodatnia oznacza, że dana osoba poprawnie klasyfikuje większą liczbę ataków niż połączeń normalnych. Aby wybrać pasujące osobniki, ustawiliśmy wartość progową wynoszącą
0.95. W ten sposób wszystkie osobniki, które mają wynik fitness > 0,95, są wybierane do produkcji kolejnych pokoleń i są uważane za sprawne. Konstrukcja funkcji fitness jest taka, że jest ona ukierunkowana na osobniki, które poprawnie klasyfikują tylko połączenia ataku, ponieważ taki jest cel. W idealnej sytuacji wybieramy 60 najlepiej dopasowanych osobników, które są poddawane krzyżowaniu i tworzą następne pokolenie. Jeśli jednak pojawi się sytuacja, w której liczba sprawnych osobników jest mniejsza niż 60, powielamy tyle samo sprawnych osobników, aby utworzyć populację macierzystą liczącą 60 osobników. Aby uzupełnić zestaw 100 osobników w populacji, 40 najlepszych potomków z 60 jest duplikowanych.

4.3 KRZYŻOWANIE I MUTACJA

Dopasowane osobniki wybrane na podstawie ich wyniku sprawności są teraz poddawane krzyżowaniu w celu wygenerowania nowych reguł lub hipotez w celu prawidłowej klasyfikacji połączeń ataków, które mogą nie występować w zestawie danych szkoleniowych. Zaimplementowaliśmy tylko krzyżowanie jednopunktowe. Przykład działania tej operacji jest następujący: Znaki pogrubione zostaną skrzyżowane. "#" jest separatorem oznaczającym różne pola.

String1: **000#0256#1169**; String2: **010#1000#1459**
Dziecko1: 101#256#1459; Dziecko2: **000#000#1169**.

Do krzyżowania dochodzi tylko na granicy separacji dwóch p ó l , a nie między nimi. Zostało to zrobione w celu zapewnienia, że powstałe potomstwo ma wartości, które utrzymują zakres dopuszczalnych wartości dla każdego pola. Aby przeszukać całą przestrzeń hipotez, mutujemy również wyniki potomne w celu wygenerowania nowych reguł lub hipotez. Zastosowaliśmy strategię mutacji pojedynczego pola, w której dowolne pole w potomstwie jest mutowane. Współczynnik mutacji został ustawiony na 1%, tj. na 100 osobników w populacji tylko 1 osobnik zostanie poddany mutacji.

5. KONFIGURACJA EKSPERYMENTALNA

4.1 CEL

Jak wspomniano wcześniej, zakres naszego eksperymentu koncentrował się na generowaniu klasyfikatorów lub reguł dla sześciu typów ataków należących do dwóch różnych klas. Zbiór treningowy zawiera maksymalną liczbę połączeń typu ataku smurf, a dokładniej 280 790. Liczba pozostałych połączeń wynosi: 107201 połączeń neptune, 21 połączeń land, 15 połączeń satan, 30 połączeń ipsweep i 15 połączeń portsweep. W związku z tym chcieliśmy stworzyć regułę, która może sklasyfikować wszystkie te połączenia przy minimalnym odsetku wyników fałszywie dodatnich. Chociaż wolelibyśmy rozszerzyć naszą implementację na wszystkie typy ataków i cechy połączeń, ogromna złożoność czasowa algorytmu, bardzo duże zbiory danych i brak czasu ograniczyły nas.

4.2 NARZĘDZIA

Do naszej implementacji wykorzystaliśmy bibliotekę GALIB C++, specjalnie dostosowaną do tworzenia algorytmów automatycznych [12]. Ze względu na dużą przestrzeń wyszukiwania hipotez i wysoką złożoność czasową, chcieliśmy użyć narzędzia lub biblioteki o wysokiej wydajności i szybkości obliczeniowej. Po kompleksowej analizie wielu narzędzi zdecydowaliśmy się na GALIB, ponieważ jest to biblioteka C++, szeroko stosowana przez innych badaczy i dobrze udokumentowana. Do wykonania programu komputerowego użyliśmy komputera Dell z systemem LINUX, procesorem Pentium 4, 120 GB miejsca na dysku twardym i 1 GB pamięci RAM.

4.3 PRZESTRZEŃ WYSZUKIWANIA HIPOTEZ

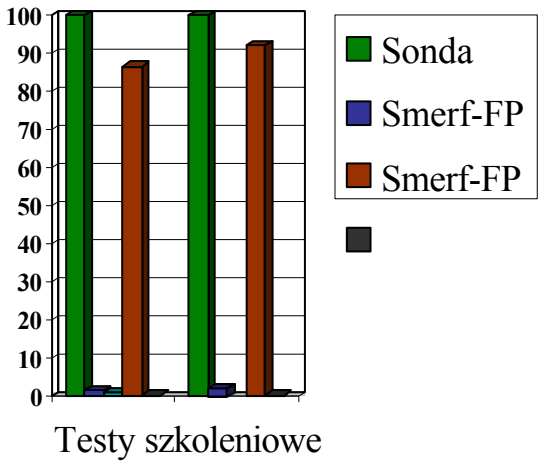
W tym eksperymencie ograniczyliśmy przestrzeń wyszukiwania hipotez do ośmiu najważniejszych dziedzin, które mogliśmy zidentyfikować. Chociaż możemy rozszerzyć naszą przestrzeń wyszukiwania do wszystkich 41 pól, b ę d z i e t o jednak wymagało obliczeń trwających wiele godzin. Dlatego w naszym eksperymencie ograniczyliśmy przestrzeń wyszukiwania tylko do ośmiu ważnych dziedzin. Pola te wybraliśmy na podstawie heurystycznej analizy danych szkoleniowych w celu zidentyfikowania potencjalnych pól które wydawały się unikalne dla określonego typu ataku. Pola te dostarczają informacji o typie protokołu, typie usługi, fladze polecenia w sesji kontrolnej ftp cechy połączenia. Z naszej analizy danych treningowych wynika, że pola te są jednymi z decydujących, które mogą pomóc w identyfikacji ataku od normalnego połączenia. W kolejnych eksperymentach zamierzamy rozszerzyć naszą implementację o więcej pól. Wielkość populacji dla każdego pokolenia zawiera 100 osobników.

4.4 OPERATORY i PARAMETRY GENETYCZNE

Mówimy, że osobnik pasuje do typu ataku, gdy wszystkie osiem pól, które tworzą naszą przestrzeń wyszukiwania osobnika, pasuje do tych z połączenia ataku. Współczynnik krzyżowania został ustawiony na 0,6, tj. biorąc pod uwagę 100 osobników w dowolnej populacji, 60 najlepiej dopasowanych osobników zostanie wybranych na podstawie najwyższego wyniku sprawności i zostanie poddanych krzyżowaniu w celu utworzenia potomstwa. Ponieważ obecnie badamy tylko osiem pól, krzyżowanie odbywa się tylko na tych polach. Spośród 60 rodziców wybieranych jest 40 najlepszych, aby uzupełnić populację do 100 osobników. W ten sposób najlepiej dopasowani rodzice uczestniczą również w kolejnych pokoleniach. Współczynnik mutacji został ustalony na 1%, co oznacza, że tylko 1 osobnik z populacji o wielkości 100 ulega zmianie w jednym z ośmiu pól, jak wyjaśniono w sekcji 4.3.

5. WYNIKI

Na podstawie powyższego eksperymentu udało nam się stworzyć regułę, która z powodzeniem sklasyfikowała wszystkie z 280 790 połączeń atakujących typu smurf. Oprócz tego sklasyfikowała również 410 normalnych połączeń jako atak. Współczynnik wyników fałszywie dodatnich wynosi zatem około 0,08%**.** W całym zestawie danych testowych zestaw reguł smerfów poprawnie sklasyfikował 2 807 880 połączeń, a odsetek wyników fałszywie dodatnich wyniósł 0,17%. Zestaw reguł, który klasyfikuje ataki typu Probe, był w stanie poprawnie sklasyfikować 52 połączenia ataków typu Probe w zestawie danych treningowych, z 60 ataków typu Probe. Zestaw reguł dla sond na całych danych testowych wykazał n a s t ę p u j ą c e w y n i k i : całkowita liczba ataków sond = 38 786, całkowita poprawnie sklasyfikowana liczba = 35 829. Dokładność procentowa = 92,3%. Są to zachęcające wyniki, biorąc pod uwagę, że użyliśmy tylko 8 pól. Wszystkie te wyniki przedstawiono na rysunku 2.



(błąd lub normalne połączenie), czas trwania wysłanych bajtów, czas trwania połączenia, % połączeń z różnymi hostami, liczba operacji na plikach kontroli dostępu i liczba połączeń wychodzących.

Rysunek 2: Wykres przedstawiający procentową dokładność bazy reguł dla dwóch klas ataków, a mianowicie smurf (DOS) i ataków sondujących, wraz ze wskaźnikami fałszywie dodatnimi oznaczonymi przez FP.

6. WNIOSKI

Nasze wyniki były bardzo zachęcające. Byliśmy w stanie wygenerować regułę wykorzystującą zasady ewolucji w GA, aby sklasyfikować wszystkie typy etykiet ataków smerfów w zestawie danych treningowych. Nasz współczynnik wyników fałszywie pozytywnych jest również dość niski i wynosi 0,2%, a współczynnik dokładności sięga 100%. Wyniki te dały nam impuls do rozszerzenia naszego programu i zastosowania go do wyszukiwania wszystkich pól w połączeniach. Mamy nadzieję, że poprawi to wydajność GA, biorąc pod uwagę, że obecnie jesteśmy w stanie sklasyfikować wszystkie połączenia typu ataku smerfowego na podstawie dopasowania tylko ośmiu pól.

Planowaliśmy zastosować Boosting w naszym systemie, aby zwiększyć jego wydajność. Chociaż boosting może być skutecznie stosowany w celu zminimalizowania błędu w IDS, obecnie go nie wdrożyliśmy. Ogólna złożoność tego projektu była sporym wyzwaniem, z którym musieliśmy sobie poradzić, aby umożliwić nam wdrożenie Boostingu.

7. PRZYSZŁA PRACA

Oprócz przeszukiwania całej przestrzeni hipotez składającej się z 41 cech, istnieją pewne obszary, których zbadanie może poprawić wydajność naszego systemu. Obecnie nasze kryterium kondycji do wybierania pasujących osób opiera się na ośmiu polach, które dokładnie pasują do połączenia ataku. Można stworzyć metrykę opartą na statystykach, aby znaleźć zakres wartości dla każdego z tych ośmiu pól, które najlepiej klasyfikowałyby etykiety ataków. Można również użyć technik eksploracji danych, aby zidentyfikować najbardziej znaczące pola, które mogą identyfikować połączenia ataku. Dodałoby to więcej naukowego uzasadnienia do przestrzeni poszukiwań algorytmu.

PODZIĘKOWANIA

Chcielibyśmy wyrazić nasze podziękowania i wdzięczność profesorowi Bryanowi Pardo (profesor EECS, Northwestern University) za nasze zabawne i wzbogacające doświadczenie w prowadzonym przez niego kursie uczenia maszynowego. Dziękujemy również naszym przyjaciołom z klasy, a mianowicie Jinyu, Alexowi, Nickowi i innym, którzy zrecenzowali nasz artykuł i przekazali nam cenne sugestie.

ODNIESIENIA

[1] Amor et al. Naïve Bayes vs Decision Trees in intrusion detection systems, *ACM Symposium on Applied Computing*, 2004.

[2] Wei Li. Wykorzystanie algorytmu genetycznego do wykrywania włamań do sieci. *Proceedings of the United States Department of*

[3] KDDCUP 1999 http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[4] Vladimir, M., Alexei, V., and Ivan, S. The MP13 approach to the KDD'99 classifier learning contest. *SIGKDD Explorations, 2000 ACM SIGKDD*. *1*(2). Styczeń 2000.

[5] Crosbie, M., and Spafford, G. Zastosowanie programowania genetycznego do wykrywania włamań. In *Proc.1995 AAAI Symposium on Genetic Programming*, pp. 1-8.

[6] Pfahringer, B. Winning the KDD99 Classification Bagged boosting. *SIGKDD Explorations, 2000 SIGKDD*. *1*(2), pp. 65-66, styczeń 2000.

[7] Dasgupta, D., and Gonzalez, F. A. An intelligent decision support system for intrusion detection and response. In *Proc. Int'l Workshop on Mathematical Methods, Models and Arch.For Computer Networks Security*, pp. 1-14, 2001.

[8] Ghosh, A., and Schwartzbard, A. A study in using neural networks for anomaly and misuse Detection. *8th USENIX Security Symposium*, pp. 141-151, 1999.

[9] Saniee M., Habibi J., Lucas C. Intrusion detection using a fuzzy genetics-based learning algorithm. *Journal of Network and Computer Applications*, 30(1), pp. 414 - 428. Styczeń 2007

[10] *Konferencja szkoleniowa Energy Cyber Security Group 2004*, Kansas City, Kansas, 24-27 maja 2004 r.

[11] Adithya Chittur. Model generation for an intrusion detection system using genetic algorithms, Ossing, NY. 2001

[12] GALIB, biblioteka c++ komponentów GA, http://lancet.mit.edu/galib-2.4/