Информационные технологии

УДК 004.042

Обнаружение сетевых атак с помощью генетически создаваемых конечных автоматов

В. П. Фраленко

Федеральное государственное бюджетное учреждение науки
Институт программных систем им. А.К. Айламазяна Российской академии наук
ул. Петра I, д. 4а, с. Веськово, Переславский район, Ярославская область, Россия, 152021

Представлены два метода обнаружения сетевых атак с помощью генетически создаваемых конечных автоматов с а) действиями на переходах и б) с выделенными состояниями. В основе первого метода лежит модель «флиба», способная предсказывать изменения сетевой активности на основе поступательного анализа сетевых записей в формате KDD-99. Второй метод является адаптацией классического конечного автомата.

Ключевые слова: конечный автомат, сетевая атака, генетический алгоритм, мутация, скрещивание, полнота, точность, «флиб», состояние.

Введение

Для моделирования и обнаружения таких динамических объектов как сетевые атаки может быть с успехом использована общая теория конечных автоматов (КА). Конечный автомат можно охарактеризовать как устройство, имеющее входные и выходные каналы и находящееся в каждый из дискретных моментов времени, называемых тактами, в одном из конечного числа состояний. Далее будут рассмотрены два предложенных автором метода на основе генетически создаваемых КА с а) действиями на переходах и б) с выделенными состояниями.

Распознающий конечный автомат — это пятёрка $\{Q, A, \delta, \lambda, F\}$, где:

- Q — множество конечных состояний автомата;
- A — алфавит автомата (конечное множество входных U, выходных V и промежуточных символов или букв);
- δ — функция переходов, задающая отображение вида $\delta : Q \times A \to Q$ (переход из одного состояния в другое под воздействием символов входного алфавита);
- λ — функция выходов, её вид зависит от типа конечного автомата;
- F — множество выделенных состояний, являющееся подмножеством множества Q. Это могут быть, например, состояние нормального функционирования сети, состояние атаки или состояние, соответствующее конкретному классу проводимой атаки. В случае метода (а) множество F пустое.

Создаваемые КА могут быть как детерминированными, в которых для каждой последовательности входных символов существует лишь одно состояние, так и недетерминированными. В первом случае характер сетевой активности всегда однозначно определяется, во втором распознающий автомат получается из объединения нескольких автоматов, распознающих разные виды сетевой активности (при этом возможны ситуации с неоднозначным ответом — в случаях, когда получаются разные значения функций выходов, достигается сразу несколько конечных состояний, что решается введением комитета большинства).

Проблема всех существующих решений в обнаружении сетевых атак конечными автоматами и графами атак в том, что все они строятся эвристически — на основании знаний экспертов. Представленные в научных работах автоматы и графы очень абстрактны, условия переходов зачастую формализованы на естественных языках [1–8]. При этом появление новых модификаций известных атак уже не укладывается в имеющиеся модели. Приходится строить новый автомат/граф. В случае же генетического обучения необходимо лишь расширить

обучающую выборку. Пройдёт несколько мутаций/скрещиваний и распознающий автомат получит набор необходимых состояний и переходов.

1. Модель «флиба» на основе конечного автомата

Генетические алгоритмы являются одним из эффективных способов для генерации автоматов. Исследования, направленные на разработку более эффективных генетических алгоритмов для построения автоматов, весьма актуальны. Наиболее известны в этом плане работы А. А. Шалыто, в том числе посвящённые «флибам» Л. Фогеля [9]. В проведённых экспериментах Фогель моделировал поведение простейшего живого существа, названного им как «флиб». Оно способно предсказывать изменения параметра среды, обладающего периодичностью. Это существо моделировалось конечным автоматом с действиями на переходах — автоматом Мили [10]. В качестве среды выступала последовательность символов над двоичным алфавитом, например, (1111010010). На вход автомата в каждый момент времени подавалось битовое значение параметра окружающей среды. Автомат формировал значение выходной переменной — возможное значение рассматриваемого параметра в следующий момент времени.

Движение по автомату заканчивается в трёх случаях: не найдено ни одного применимого перехода, закончилась цепочка входных символов или произошло попадание в выделенное состояние. Последний вариант возможен лишь для метода (б). В методе (а) сетевая активность рассматривается вместо упомянутой ранее среды, а выходная переменная — флаг, указывающий на то, какой эта активность является — позитивной/нейтральной (флаг равен нулю) или разрушительной/враждебной (от единицы до бесконечности, флаг выступает как идентификатор класса сетевой атаки). В режиме определения факта атаки флаг может быть равен лишь нулю и единице.

2. Формат входных символов

Обучающая и тестовая выборки, использованные в экспериментах, состояли из сетевых записей, формирующих три класса: «норму», DoS-атаку и Nmap (сканирование портов). В качестве формата входных символов была использована дискретизированная версия формата базы KDD-99 [11]. Признаки, для которых характерен диапазон от 0 до 1, как и признаки, для которых нет супремума (верхней границы), преобразованы в новые целочисленные. Диапазон последних от 0 до некоторого небольшого числа, представляющего собой ограничение на число возможных условий перехода. В число таких дискретизированных признаков попали следующие:

- продолжительность соединения (duration);
- количество информации, переданное от источника к приёмнику в ходе сессии (src_bytes);
- количество информации, переданное от приёмника к источнику в ходе сессии (dst_bytes);
- число «неправильных» (т.е. с некорректными значениями SN и ACK_SN) сегментов (wrong_fragment);
- число пакетов с установленным битом URG (urgent);
- количество входов в системные директории, запуск системных программ, создание программ (hot);
- количество неудачных попыток регистрации в системе (num_failed_logins);
- количество ошибок типа «не найден» в соединении (num_compromissed);
- количество пользователей, зарегистрировавшихся в системе с правами суперпользователя (num_root);
- количество созданных файлов с момента последнего вызова (num_file_creations);

- количество входов в систему, а также получений доступа к оболочке интерпретатора команд текстового режима (num_shells);
- количество операций с управляющими файлами операционной системы (num_access_files);
- количество исходящих команд в FTP-сессии обмена файлами (num_outbound_cmds);
- количество соединений с текущим хостом за последние две секунды сетевой активности (count);
- количество обращений к текущему сервису за последние две секунды сетевой активности (srv_count);
- другие статистические признаки по последним двум секундам сетевой активности (same_srv_rate, diff_srv_rate, srv_diff_host_rate, srv_serror_rate, serror_rate, rerror_rate, srv_rerror_rate);
- количество соединений с текущим хостом за последние 100 соединений (dst_host_count);
- количество обращений к текущему сервису за последние 100 соединений (dst_host_srv_count);
- другие статистические признаки за последние 100 соединений (dst_host_rerror_rate, dst_host_srv_rerror_rate, dst_host_same_srv_rate, dst_host_diff_srv_rate, dst_host_same_src_port_rate, dst_host_serror_rate, dst_host_srv_diff_host_rate, dst_host_srv_serror_rate) [12].

3. Фитнес-функции и создание эффективного распознающего автомата

Синтезируемый автомат должен эффективно распознавать «норму» и все классы атак из обучающей выборки. В связи с этим были рассмотрены следующие варианты фитнес-функций:

а) минимальная полнота/точность для классов обучающей выборки (оказалась неэффективной из-за того, что в начале обучения практически все автоматы плохо распознают какой-нибудь из классов, что приводит к невозможности селекции);

б) средняя полнота/точность для классов обучающей выборки (оказалась неэффективной, так как в победителях зачастую оказывались автоматы с низким распознаванием одного/нескольких классов и отличным распознаванием всех остальных классов);

в) усреднённая сумма фитнес-функций (а) и (б) (получаются автоматы, имеющие хорошую минимальную и среднюю точность/полноту).

Процесс создания эффективного распознающего автомата заключается в следующем:

а) создаётся популяция из C_A автоматов с небольшим числом состояний (для метода (б) дополнительно добавляются выделенные состояния: для «нормы» и атаки, либо для «нормы» и несколько отдельных состояний для разных классов атак), в экспериментах C_A было взято равным 250; при этом максимальное стартовое число переходов не превышает задаваемого порога (порядка 100);

б) производится оценка текущего поколения автоматов (если качество удовлетворительное, то возможен останов работы) и отбор не более C_{next} и не менее C_{best} автоматов (остальные автоматы затем удаляются — турнирный подход [13], в экспериментах C_{next} было взято равным 30) с учётом следующего условия: значение фитнес-функции соседних автоматов должно отличаться на величину не менее ε, вычисляемую адаптивно на основании информации о значении фитнес-функции (статусе) лучших C_{next} автоматов: $\varepsilon = \eta(1 - S_1 + S_{C_{next}})/C_{next}$, где η — коэффициент адаптивности (в экспериментах было взято равным 0.05), S_1 — статус лучшего из автоматов и $S_{C_{next}}$

— статус автомата с номером C_{next} (такой подход позволяет исключить попадание в локальные минимумы);

в) для выбранных C_{next} автоматов либо происходит скрещивание пар взятых случайным образом автоматов, либо отдельные автоматы путём одной из описанных в следующем разделе мутаций производят потомков; вероятности появления мутаций и скрещиваний и ограничение на число мутаций на автомат задаются настройкам;

г) получившиеся в результате мутаций и скрещиваний автоматы добавляются в новую популяцию до тех пор, пока её размер не станет снова равен C_A (численность популяции не изменяется от одной эпохи к другой), при этом некоторое малое число лучших автоматов — C_{best} (в эксперименте было взято равным 5), входящих в выборку, полученную на шаге (б), переносится в новую популяцию без изменений, что позволяет не терять лучшие особи (принцип элитизма [14]);

д) переход на шаг (б).

4. Мутации и перекрёстные скрещивания

При перекрёстном скрещивании новый автомат получается из пары уже существующих: в него добавляются состояния то из одного, то из другого автомата родительского автомата (выбор случайный). Для отдельных автоматов применимы следующие виды мутаций:

- замена начального состояния (в случае, если число состояний больше единицы);
- удаление перехода;
- добавление перехода;
- замена направления перехода;
- замена условия перехода;
- замена действия на переходе (под действием понимается флаг сетевой активности).

Последняя мутация возможна лишь в случае с автоматами без выделенных состояний, то есть в случае метода (а).

Скрещивание наиболее приспособленных особей приводит к тому, что исследуются наиболее перспективные участки пространства поиска. В конечном итоге популяция будет сходиться к оптимальному решению задачи. В процессе мутаций и скрещиваний, вероятности появления которых задаются настройками, возможно появление автоматов с недоступными состояниями (в которые невозможно попасть) и незадействованными переходами (для которых на обучающей выборке ни разу не выполнилось условие перехода). Если в процессе генетического обучения от них есть несомненная польза, так как они могут становиться активными, то после обучения необходимости в них нет.

5. Эксперимент с автоматами с действиями на переходах

После очистки от недостижимых состояний и незадействованных переходов автомат сохраняется в формате Graphviz [15] на языке DOT, грамматика которого соответствует форме Бэкуса–Наура [16]. Пример автомата приведён на рис. 1.

Как оказалось, получившиеся автоматы (время обучения на выборке из полумиллиона записей сетевой активности порядка 10 часов или 1000 эпох обучения) достаточно хорошо распознают как обучающую, так и тестовую выборки. Трудности возникли лишь с DoS-атаками (табл. 1). Если же определять лишь факт атаки, то настроенный за аналогичное время автоматный классификатор справляется с поставленной задачей с приемлемым качеством (табл. 2). Полученные решения характеризуются сверхвысокой скоростью работы (более 500 тысяч

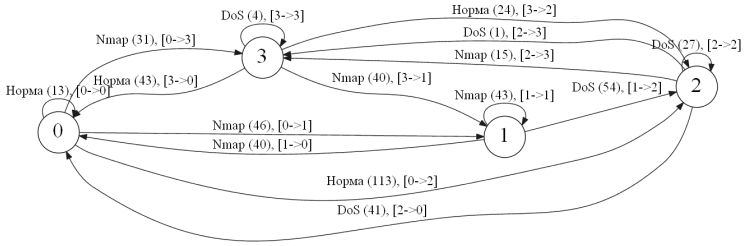

Рис. 1. Пример распознающего автомата с действиями на переходах

записей о сетевой активности в секунду, что на порядок быстрее известных решений на основе нейронных сетей). Одно из возможных расширений описанного алгоритма — построение комитета большинства, использующего полученный в результате работы автомата набор флагов сетевой активности.

Таблица 1

Распознавание тестовой и обучающей выборок (определение нормы и классов атак)

	Обучающая выборка			Тестовая выборка		
Показатель	**норма**	**Nmap**	**DoS**	**норма**	**Nmap**	**DoS**
точность	0.9136	0.8881	0.9877	0.9964	0.9790	0.9536
полнота	0.9977	0.9961	0.5824	0.9999	0.9968	0.6109

Таблица 2

Распознавание тестовой и обучающей выборок (определение факта атаки)

	Обучающая выборка		Тестовая выборка	
Показатель	**норма**	**атака**	**норма**	**атака**
точность	0.9638	0.9994	0.9995	0.9967
полнота	0.9732	0.9992	0.9930	0.9998

6. Эксперимент с автоматами с выделенными состояниями

Пример автомата с выделенными состояниями приведён на рис. 2. Как оказалось, в задаче определения «нормы» и классов атак автомат с выделенными состояниями близок по результатам к автомату с действиями на переходах (табл. 3). Однако в задаче определения факта атаки пара выделенных состояний позволила значительно превзойти результат автомата с действиями на переходах (табл. 4). В частности, на тестовой выборке обеспечивается 100%-я точность распознавания «нормы». При этом за счёт меньшего числа переходов получившиеся автоматы обрабатывают порядка 700 тысяч записей о сетевой активности в секунду.

Таблица 3

Распознавание тестовой и обучающей выборок (определение нормы и классов атак)

	Обучающая выборка			Тестовая выборка		
Показатель	**норма**	**Nmap**	**DoS**	**Норма**	**Nmap**	**DoS**
точность	0.9164	0.8882	0.9921	0.9959	0.9790	0.9744
полнота	0.9979	0.9967	0.5834	0.9999	0.9973	0.6109

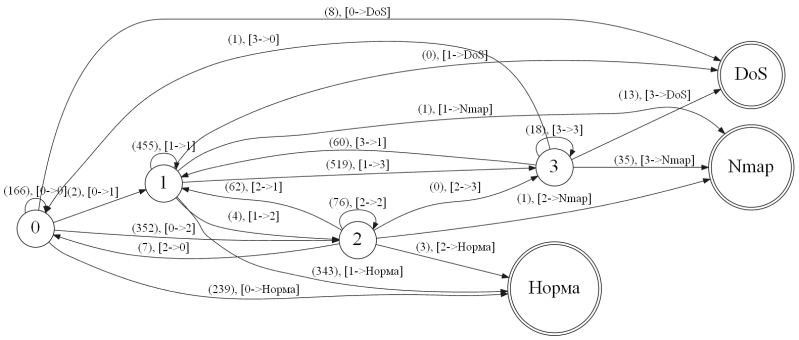

Рис. 2. Пример распознающего автомата с выделенными состояниями

Таблица 4

Распознавание тестовой и обучающей выборок (определение факта атаки)

	Обучающая выборка		Тестовая выборка	
Показатель	**норма**	**атака**	**норма**	**атака**
точность	0.9848	0.9997	1.0000	0.9995
полнота	0.9844	0.9997	0.9989	1.0000

Заключение

Таким образом, можно говорить о появлении нового класса алгоритмов обнаружения сетевых атак с помощью генетически обучаемых конечных автоматов, которые характеризуются

- приемлемым временем, затрачиваемым на обучение,
- отсутствием необходимости в экспертных знаниях,
- молниеносной скоростью обнаружения сетевых атак,
- возможностью дообучения как на старой, так и расширенной обучающих выборках,
- интуитивно понятным принципом работы и простотой программирования.

Оба предложенных метода справляются с задачей обнаружения факта атаки, однако возникают проблемы с определением её класса (низкая полнота). Для повышения качественных показателей предлагается сначала определять факт атаки автоматом с выделенными состояниями, а затем определять класс обнаруженной атаки с помощью комитета классификаторов.

Исследования проводятся в рамках работ по Государственному контракту № 07.514.11.4048 по теме «Разработка интеллектуальных методов автоматизированного обнаружения и предотвращения распределённых сетевых атак и их реализация в современных системах облачных вычислений» (шифр заявки «2011–1.4–514–017–004»).

Разработанное программное обеспечение — закрытое (исходные коды не распространяются), используется только в проектах ИПС им. А.К. Айламазяна РАН.

Литература

1. *Городецкий В. И., Котенко И. В., Маньков Е. В.* Моделирование распределенных атак на компьютерные сети. — СПб. — С. 56–57. [*Gorodeckiyj V. I., Kotenko I. V., Manjkov E. V.* Modelirovanie raspredelennihkh atak na kompjyuternihe seti. — SPb. — S. 56–57.]
2. *Галатенко А. В.* Об автоматной модели защищенных компьютерных систем. — Т. 4. — М. — С. 263–270. [*Galatenko A. V.* Ob avtomatnoyj modeli zathithennihkh kompjyuternihkh sistem. — T. 4. — M. — S. 263–270.]

3. Automated Generation and Analysis of Attack Graphs / O. Sheyner, S. Jha, J. Wing et al. — Oakland, USA.
4. *Колегов Д. Н.* Проблемы синтеза и анализа графов атак // Вестник ТГУ. Приложение. — 2007. — № 23. — С. 180–188. [*Kolegov D. N.* Problemih sinteza i analiza grafov atak // Vestnik TGU. Prilozhenie. — 2007. — No 23. — S. 180–188.]
5. *Lippmann R., Ingols K., Scott C.* Evaluating and Strengthening Enterprise Network Security Using Attack Graphs. — `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.92.3063&rep=rep1&type=pdf`.
6. *Lippmann R. P., Ingols K. W., Piwowarski K.* Practical Attack Graph Generation for Network Defense. — `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.112.8029&rep=rep1&type=pdf`.
7. *Sheyner O.* Scenario Graphs and Attack Graphs: Ph.D. thesis / Carnegie Mellon University. — Pittsburgh, USA, 2004.
8. *Степашкин М. В., Котенко И. В., Богданов В. С.* Интеллектуальная система анализа защищенности компьютерных сетей. — М.: Физматлит. [*Stepashkin M. V., Kotenko I. V., Bogdanov V. S.* Intellektualjnaya sistema analiza zathithennosti kompjyuternihkh seteyj. — M.: Fizmatlit.]
9. *Фогель Л., Оуэнс А., Уолш М.* Искусственный интеллект и эволюционное моделирование. — М.: Мир, 1969. [*Fogelj L., Ouehns A., Uolsh M.* Iskusstvennihyj intellekt i ehvolyucionnoe modelirovanie. — M.: Mir, 1969.]
10. *Шалыто А. А.* SWITCH-технология. Алгоритмизация и программирование задач логического управления. — СПб.: Наука, 1998. [*Shalihto A. A.* SWITCH-tekhnologiya. Algoritmizaciya i programmirovanie zadach logicheskogo upravleniya. — SPb.: Nauka, 1998.]
11. Fifth ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. — `http://www.sigkdd.org/kdd1999/`.
12. *Kayacik H. G., Zincir-Heywood A. N., Heywood M. I.* Selecting Features for Intrusion Detection: A Feature Relevance Analysis on KDD 99 Intrusion Detection Datasets. — St. Andrews, Canada.
13. *Miller B., Goldberg M.* Genetic Algorithms, Tournament Selection, and the Effects of Noise. — Vol. 3. — Pp. 193–212.
14. *De Jong K.* An Analysis of the Behavior of a Class of Genetic Adaptive Systems: Ph.D. thesis / University of Michigan. — Ann Arbor, USA, 1975.
15. Graphviz. Graph Visualization Software. — `http://www.graphviz.org`.
16. The DOT Language. Graph Visualization Software. — `http://www.graphviz.org/content/dot-language`.

UDC 004.042

Intrusion Detection using Genetically Generated Finite Automata

V. P. Fralenko

Organization of Russian Academy of Sciences Program System Institute of RAS
4a, Peter I str., Veskovo, Pereslavl-Zalessky, Yaroslavl Region, Russia, 152021

Two new methods for detecting network attacks using genetically generated finite automata with a) the transitions actions, and b) with the selected states are presented. The first method is based on the "flib" model that can predict changes in network activity on the basis of progressived analysis of network records in the KDD-99 format. The second method is an adaptation of a classical finite automata.

Key words and phrases: finite automata, network attack, genetic algorithm, mutation, crossover, recall, precision, "flib", state.