Для того, чтобы получить **.exe** файл, нужно провести компоновку объектного файла. В заголовке **.exe** файла можно увидеть дополнительную секцию **.rdata.** Именно в этой секции находится строчка со значением “Hello, World!”

Таким образом я продемонстрировал, что в программировании ничего не работает просто так. Даже за самой минимальной программой лежит большое количество “прослоек”, которые работают не идеально и требуют постоянных доработок и оптимизации.

```
Dump of file Source.obj

File Type: COFF OBJECT

COFF SYMBOL TABLE
000 01046992 ABS    notype       Static       | @comp.id
001 80000191 ABS    notype       Static       | @feat.00
002 00000000 SECT1  notype       Static       | .drectve
    Section length   2F, #relocs    0, #linenums    0, checksum        0
004 00000000 SECT2  notype       Static       | .debug$S
    Section length   78, #relocs    0, #linenums    0, checksum        0
006 00000000 SECT3  notype       Static       | .text$mn
    Section length   14, #relocs    2, #linenums    0, checksum F829D91C
008 00000000 SECT4  notype       Static       | .text$mn
    Section length    A, #relocs    1, #linenums    0, checksum 71A05264, selection    2 (pick any)
00A 00000000 SECT5  notype       Static       | .text$mn
    Section length   29, #relocs    2, #linenums    0, checksum 2B25B17F, selection    2 (pick any)
00C 00000000 SECT6  notype       Static       | .text$mn
    Section length   3A, #relocs    2, #linenums    0, checksum CAE6D625, selection    2 (pick any)
00E 00000000 SECT4  notype ()    External     | ___local_stdio_printf_options
00F 00000000 UNDEF  notype ()    External     | ___acrt_iob_func
010 00000000 UNDEF  notype ()    External     | ___stdio_common_vfprintf
011 00000000 SECT5  notype ()    External     | __vfprintf_l
012 00000000 SECT6  notype ()    External     | _printf
013 00000000 SECT3  notype ()    External     | _main
014 00000008 UNDEF  notype       External     | ?_OptionsStorage@?1??__local_stdio_printf_options@@9@9
rintf_options'::`2'::_OptionsStorage)
015 00000000 SECT7  notype       Static       | .data
    Section length    D, #relocs    0, #linenums    0, checksum AF57950C
017 00000000 SECT7  notype       Static       | $SG7440
018 00000000 SECT8  notype       Static       | .chks64
    Section length   40, #relocs    0, #linenums    0, checksum        0
```

Рисунок 1 – Символьная таблица

ВЫЯВЛЕНИЕ DNS ТУННЕЛИРОВАНИЯ В КОМПЬЮТЕРНОЙ СЕТИ МЕТОДОМ POMDP

Бубнов Я.В.

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Иванов Н.Н – к.физ-м.н.,доцент

В статье предлагается способ выявления целевой кибератаки, в частности DNS туннелирования в компьютерной сети. Предложенный метод базируется на использовании параметризированного частично-наблюдаемого марковского процесса принятия решений.

DNS туннелирование — это техника инкапсуляции произвольных данных через протокол системы доменных имен. В течение последних лет атаки, связанные с кражей информации из платежных систем, так или иначе эксплуатировали данную уязвимость протокола. Ярким примером может служить программа FrameworkPOS [1], с помощью которой в 2014-2015 годах осуществлялась масштабная кража информации о кредитных картах с платежных систем. Основная трудность, связанная с выявлением DNS туннелирования, заключается в сложности дифференциации туннелируемого трафика от обычного.

В виду того, что отключение системы DNS в компьютерных сетях представляется принципиально невозможным, требуются более интеллектуальные способы решения задачи – обнаружения DNS туннелирования.

За последние годы исследования проблемы обнаружения DNS туннелирования в компьютерных сетях предложено некоторое количество подходов в построении детекторов. Например, в статье [2] Надлер, Аминов и Шабтай решают проблему обнаружения туннелирования

методом бинарной классификации сетевого трафика, добившись в своих экспериментах 99-процентной чувствительности детектора. В статье [3] автор предлагает использовать многослойную feedforward нейронную сеть для классификации трафика по различным методикам туннелирования, что предоставляет возможность дифференциации политик блокировки в зависимости от подвида атаки. Автор демонстрирует 85-процентную точность детектора на тестовом наборе данных.

На практике, эффективность вышеупомянутых способов обнаружения DNS туннелирования в значительной степени зависят от исходного, обучающего набора данных. Процесс обучения модели, в свою очередь, становится зависимым от типов туннелирования, масштабов и конфигурации сети.

С целью решения поставленной проблемы: детектирования DNS туннелирования при использовании неидеальных детекторов, в работе [4] МакКарти предлагает использовать модификацию POMDP, а именно - виртуально-распределенный POMDP. В своей работе МакКарти строит механизм принятия решения о блокировке доменного имени исходя из объема и ценности трафика, направляемого в данный домен.

Очевидные трудности, которые возникают при использовании подхода МакКарти, связаны с необходимостью анализа совокупного трафика сети, что при возрастании масштабов сети, становится трудновыполнимым. Вместо анализа совокупного трафика должен производиться локальный анализ непосредственно DNS запроса.

Для эффективного использования POMDP при обнаружении DNS туннелирования, функция вознаграждения должна быть сформулирована следующим образом. Пусть L ϵ [0, 1] — это требуемый уровень обеспечения безопасности сети. Тогда, при L = 0 достигается наименьшая степень защищенности сети, и в то же время наибольший уровень доступности системы, так как вероятность заблокировать важный для функционирования системы домен сводится к нулю. При L = 1 достигается наибольшая степень защищенности, однако вероятность ложно позитивных блокировок доменов возрастает, что сказывается на функционировании сети.

Исходя из этого, L-параметризированную POMDP модель для обнаружения DNS туннелирования можно задать следующим образом:

S = {Blocking, Passing}
A = {Analyze, Block, Pass}
O = {OK, Bad}

Где S — совокупность состояний системы: DNS запрос блокируется или пропускается системой. A — совокупность возможных действий системы: опросить детектор, заблокировать запрос или пропустить запрос. O — совокупность возможных наблюдений: детектор определил запрос как безопасный, детектор отнес запрос к классу DNS туннелей.

Путь Q — точность детектора, а L — уровень безопасности, а K — коэффициент сложности доступа к детектору, тогда функция вознаграждения R(O, A) описывается в таблице 1.

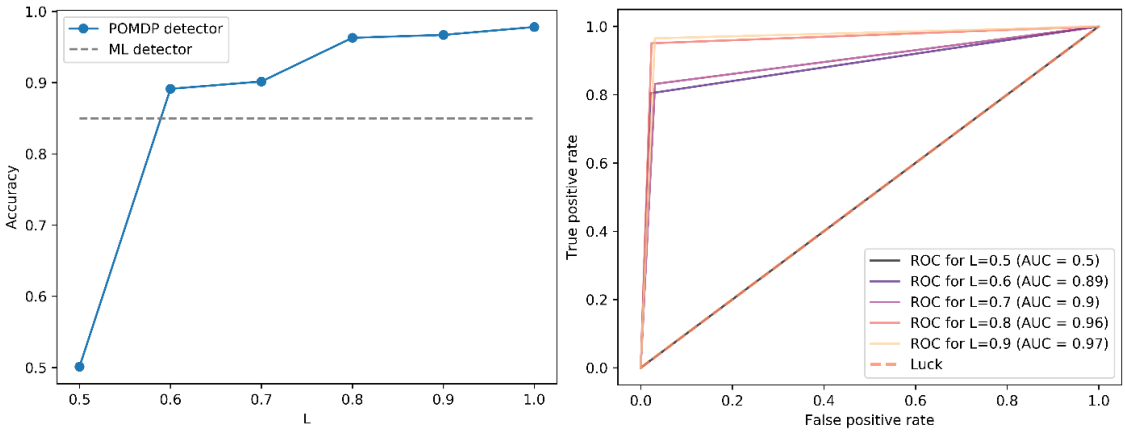
Таблица 1 - Функция вознаграждения

O	**A**	**R(O, A)**
Bad	Block	L
OK	Block	-(1-L)
Bad	Pass	1-L
OK	Pass	L
Bad	Analyze	L*K
OK	Analyze	L*K

Для проведения эксперимента использовалась тестовая установка, состоящая из клиентской и серверной станций, между которыми установлено SSH соединение через DNS туннель с помощью пакета iodine [5]. На клиентской станции установлен детектор, предложенный в [3], а также L-параметризированная POMDP система, принимающая решение о блокировке исходящих DNS запросов.

На описанной установке поставлены эксперименты со значением параметра L ϵ [0,5; 1,0]. В данном эксперименте основной задачей является поиск оптимальной политики для принятия решения о блокировке запроса. Поиск оптимальной политики осуществляется с помощью алгоритма «инкрементальной чистки», описанной в [6].

Пусть детектор с вероятностью Q = 0,85 корректно относит рассматриваемый DNS запрос к нужному классу. Тогда точность распознавания DNS туннелей в зависимости от L представлена на рисунке 1а, где штриховой линией отмечена точность распознавания исходного детектора. Качество классификации DNS туннелирования оценивается по построенным ROC-кривым, представленными на рисунке 1б.



а б

Рисунок 1а – зависимость точности модели от параметра *L*,
1б – ROC-кривые для различных значений параметра *L*

Из полученных результатов видно, что рост параметра L положительно влияет на точность распознавания DNS туннелирования. В то же время, при L < 0,6 качество распознавания резко снижается, и при L = 0,5 носит характер случайного гадания. Стоит отметь, что в этом случае, система производит меньшее количество ложно положительных ошибок, но большее количество ложно отрицательных ошибок. Это объясняется тем, что система пропускает больше подозрительного трафика с целью обеспечения доступности сети в целом.

Таким образом, в поставленном эксперименте, с помощью L-параметризированного POMDP удалось добиться увеличения точности распознавания DNS туннелирования на 10% в сравнении с детектором, представленным в статье [3].

Список использованных источников:

1. Valenzuela, I. Game Changer: Identifying and Defending Against Data Exfiltration Attempts. – Boston : Sans Institute, 2015.
2. Nadler, A. Detection of Malicious and Low Throughput Data Exfiltration Over the DNS protocol / A. Nadler, A. Aminov, A. Shabtai – Negev : Ben Gurion University of the Negev, 2017.
3. Bubnov, Y. DNS Tunneling Detection Using Feedforward Neural Network / Y. Bubnov // European Journal of Engineering Research & Science. – 2018. – Vol. 3, № 11. – P. 16-19.
4. McCarthy, S.Data Exfiltration Detection and Prevention: Virtually Distributed POMDPs for Practically Safer Networks / S. McCarthy, A. Sinha, M. Tambe, P. Manadhata – Los Angeles : University of Southern California, 2016.
5. Iodine [Электронный ресурс]. – Режим доступа: http://code.kryo.se/iodine. – Дата доступа: 03.22.2019.
6. Cassandra, A. Incremental Pruning: A Simple, Fast, Exact Method for Partially Observable Markov Decision Process / A. Cassandra, M. Littman, N. Zhang // Proceedings of the Thirteen Conference on Uncertainty in Artificial Intelligence. – 1997. – P. 54-61.

СВЕРТОЧНАЯ НЕЙРОННАЯ СЕТЬ ДЛЯ РАСПОЗНАВАНИЯ РУКОПИСНЫХ СИМВОЛОВ

Гаджиев С. С.

*Белорусский государственный университет информатики и радиоэлектроники,
г. Минск, Республика Беларусь*

Воронов А. А. – к.т.н.,доцент

Рассмотрены вопросы построения сверточной нейронной сети для распознавания рукописных символов. Определена основная модель, приведены примеры слоев нейронной сети.

В этой статье опишем структуру сверточной нейронной сети для распознавания рукописных символов. Классическая архитектура CHC представлена на рисунке 1:

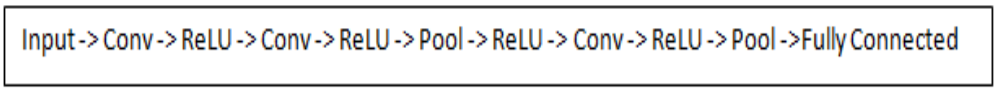


Рис. 1- Архитектура CHC

Рассмотрим, какие действия сеть будет выполнять на высоком уровне. Все фильтры можем рассмотреть, как идентификаторы свойства. Свойством могут быть кривые, прямые границы,