МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ»
имени ГУМАРБЕКА ДАУКЕЕВА
Институт Систем управления и информационных технологий
Кафедра «IT инжиниринг»

ДОПУЩЕН К ЗАЩИТЕ
Заведующий кафедрой
PhD, доцент
_____ А.А. Досжанова
«____» _____ 2020 г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

На тему: Разработка информационной системы анализирующей ЛВС на основе нейронных сетей

Специальность 6М070300 – «Информационные Системы»

Выполнил магистрант группы МИСн-18-1 _____ Кожабеков М.Т.

Научный руководитель PhD, доцент _____ Маликова Ф.У.
«____» _____ 2020 г.

Консультанты по применению
Информационных систем: ст. преп. _____ Майкотов М.Н
«____» _____ 2020 г.

Нормоконтролер: ст. преп. _____ Абсатарова Б.Р
«____» _____ 2020 г.

Рецензент: PhD, ст. науч. сотр. _____ Шаяхметова А.С.
«____» _____ 2020 г.

Алматы 2020

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ»
имени ГУМАРБЕКА ДАУКЕЕВА
Институт Систем управления и информационных технологий
Кафедра «IT инжиниринг»

Специальность 6М070300 – «Информационные Системы»

ЗАДАНИЕ
на выполнение магистерской диссертации

Магистранту Кожабеков Мадияр Талгатович

Тема проекта: Разработка информационной системы анализирующей ЛВС на основе нейронных сетей

Утверждена приказом по университету № 122 от 25.10.2018 г.

Срок сдачи законченного проекта «16» _____ июня_____ 2020 г.

Исходные данные к проекту (требуемые параметры результатов исследования (проектирования) и исходные данные объекта): Техническая документация по Python, операционная система Windows 10, данные для обучения NLS-KDD.

Перечень вопросов, подлежащих разработке в диссертационной работе, или краткое содержание диссертационной работы:

а) обзорно-аналитическая часть;
б) проектирование анализирующей ЛВС на основе нейронных сетей;
в) разработка модели обучения;

Перечень графического материала (с точным указанием обязательных чертежей): представлены 11 иллюстрации.

Основная рекомендуемая литература:

1 Саммерфилд М. Программирование на Python 3. Подробное руководство, Издательство Символ-Плюс, 2009. – 608 с.

2 E.Kesavulu., Neural Networks for Intrusion Detection and Its Applications - 2013

3 Гудфеллов Я., Бенджио И., Курвиль А. Глубокое обучение, Издательство ДМК-Пресс, 2018. – 652с.

Консультации по проекту с указанием относящихся к ним разделов работы

Раздел	Консультант	Сроки	Подпись
Программная часть	Майкотов М.Н.	05.06.2020	
Нормоконтролер	Абсатарова Б.Р.	10.06.2020	

График
подготовки диссертации

Наименование разделов, перечень разрабатываемых вопросов	Сроки представления научному руководителю	Примечание
Аналитическая часть работы	10.02.20	
Проектирование системы анализирующей ЛВС на основе нейронных сетей	13.03.20	
Разработка модели обучения	16.04.20	

Дата выдачи задания «25» __октября_____ 2020 г.

Заведующий кафедрой _____ А.А. Досжанова

Научный руководитель проекта _____ Ф.У. Маликова

Задание принял к исполнению магистрант _____ М.Т. Кожабеков

АННОТАЦИЯ

В диссертационной работе представлен анализ решений в области система обнаружения вторжений и систем обработки трафика, краткий анализ атак и анализ принципов СОВ.

Затем была предпринята попытка разработать модуль для обнаружения аномалий в сети, на основе метода нейросетевого анализа сетевой активности решаются следующие задачи:

- обнаружение активности в ЛВС;
- получение данных о перегрузках и критических .условиях. работы сети;
- обнаружение сетевых проблем и сетевых сбоев.

АННОТАЦИЯ

Диссертациялық жұмыста БАЖ және трафикті өңдеу жүйелері саласындағы шешімдерді талдау, шабуылдарды қысқаша талдау және БАЖ принциптерін талдау ұсынылған.

Содан кейін желідегі ауытқуларды анықтау үшін модульді әзірлеуге әрекет жасалды, желілік белсенділікті нейрожелілік талдау әдісі негізінде келесі міндеттер шешіледі:

- ЛВС-да белсенділікті анықтау;
- желі жұмысының шамадан тыс жүктемелері мен сыни жағдайлары туралы деректерді алу;
- желілік мәселелер мен желілік іркілістерді анықтау.

ANNOTATION

There are next principles and analysis presented in this article: the analysis of solutions in the field of Intrusion Detection System and traffic processing systems, a brief analysis of attacks and an analysis of the principles of IDS are presented.

Then an attempt was made to develop a module for detection network anomalies, based on the method of neural network analysis of network activity, the following tasks are solved:

- detection of activity in the LAN;
- receiving overload data and critical conditions network operation;
- detection networked problems and network failures.

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

IDS	Intrusion Detection System
IPS	Intrusion Prevention System
ANN	Artificial neural network
SMB	Server Message Block
DDOS	Distributed Denial of Service
KDD	Knowledge Discovery in Databases
GNG	Growing Neural Gas
ARP	Address Resolution Protocol
ICMP	Internet Control Message Protocol
PPP	Point-to-Point Protocol
FTP	File Transfer Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
DMZ	Demilitarized Zone
CPU	Central Processing Unit
IO	Information Operation
СОВ	Системы Обнаружения Вторжений
ЛВС	Лока́льная Вычисли́тельная Сеть
СПВ	Система Предотвращения Вторжений
ИНС	Искусственная нейронная сеть

Содержание

ВВЕДЕНИЕ 9
1 БЕЗОПАСНОСТЬ В ЛОКАЛЬНО ВЫЧИСЛИТЕЛЬНЫХ СЕТЯХ 10
1.1 Система обнаружения вторжений 13
1.2 Система предотвращения вторжений 25
2 ИСПОЛЬЗОВАНИЕ НЕЙРОННЫХ СЕТЕЙ В СОВ 34
2.1 Варианты подходов к решению задачи и нейросетевые решения 34
2.2 Существующие решения и предложения, основанные на нейронных сетях36
2.3 Нейронный газ 37
3 ПРОГРАММНАЯ ЧАСТЬ 43
3.1 Проектирование модуля 43
3.2 Архитектура модуля 51
ЗАКЛЮЧЕНИЕ 56
СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 58
ПРИЛОЖЕНИЕ А. Фрагменты исходных кодов 59

ВВЕДЕНИЕ

В работе Oxford Martin School 2013 года говорилось о том, что 47% всех рабочих мест может быть автоматизировано в течение следующих 20 лет. Основным драйвером этого процесса является применение искусственного интеллекта, работающего с большими данными, как более эффективной замены человеку.

Машины теперь способны решать все больше процессов, за которые раньше отвечали люди. Кроме того, делают это качественнее и во многих случаях дешевле. О том, что это значит для рынка труда, в июле этого года говорил Герман Греф, выступая перед студентами Балтийского федерального университета им. Канта:

Продолжая освещать технобудущее, команда DTI подготовила все, что необходимо знать для первого погружения в нейронные сети: как они устроены, почему все больше компаний предпочитают нейросети живым сотрудникам и какой потенциал по оптимизации различных процессов несет эта технология.

Почему нейросети вновь популярны

До 2010 года попросту не существовало базы данных, достаточно большой для того, чтобы качественно обучить нейросети решать определенные задачи, в основном связанные с распознаванием и классификацией изображений. Поэтому нейросети довольно часто ошибались: путали кошку с собакой, или, что еще хуже, снимок здорового органа со снимком органа, пораженного опухолью.

Но в 2010 году появилась база ImageNet, содержащая 15 миллионов изображений в 22 тысячах категорий. ImageNet многократно превышала объем существовавших баз данных изображений и была доступна для любого исследователя. С такими объемами данных нейросети можно было учить принимать практически безошибочные решения.

1 БЕЗОПАСНОСТЬ В ЛОКАЛЬНО ВЫЧИСЛИТЕЛЬНЫХ СЕТЯХ

Крайне важно понять, что безопасность — это не продукт, который можно купить в магазине и быть уверенным в собственной защищенности. «Безопасность» — особая комбинация как технических, так и административных мер. Административные меры также включают в себя не только бумаги, рекомендации, инструкции, но и людей. Невозможно считать свою сеть «безопасной», если вы не доверяете людям, работающим с этой сетью.

Идеальная безопасность — недостижимый миф, который могут реализовать, в лучшем случае, только несколько профессионалов. Есть один фактор, который невозможно преодолеть на пути к идеальной безопасности — это человек.

Основные цели сетевой безопасности приведены в соответствий с рисунком 1.1.

Цели сетевой безопасности могут меняться в зависимости от ситуации, но основных целей обычно три:

- целостность данных;
- конфиденциальность данных;
- доступность данных;
- рассмотрим более подробно каждую из них.

Рисунок 1.1 – Основные цели сетевой безопасности

Целостность данных. Одна из основных целей сетевой безопасности — гарантированность того, чтобы данные не были изменены, подменены или уничтожены. Целостность данных должна гарантировать их сохранность как в случае злонамеренных действий, так и случайностей. Обеспечение целостности данных является обычно одной из самых сложных задач сетевой безопасности.

Конфиденциальность данных. Второй главной целью сетевой безопасности является обеспечение конфиденциальности данных. Не все данные можно относить к конфиденциальной информации. Существует достаточно большое количество информации, которая должна быть доступна всем. Но даже в этом случае обеспечение целостности данных, особенно открытых, является основной задачей. К конфиденциальной информации можно отнести следующие данные:

- личная информация пользователей;
- учетные записи (имена и пароли);
- данные о кредитных картах;

- данные о разработках и различные внутренние документы;
- бухгалтерская информация.

Доступность данных. Третьей целью безопасности данных является их доступность. Бесполезно говорить о безопасности данных, если пользователь не может работать с ними из-за их недоступности. Вот приблизительный список ресурсов, которые обычно должны быть «доступны» в локальной сети:

- принтеры;
- серверы;
- рабочие станции;
- данные пользователей;
- любые критические данные, необходимые для работы.

Рассмотрим угрозы и препятствия, стоящие на пути к безопасности сети. Все их можно разделить на две большие группы: технические угрозы и человеческий фактор.

Технические угрозы:

- ошибки в программном обеспечении;
- различные dos- и ddos-атаки;
- компьютерные вирусы, черви, троянские кони;
- анализаторы протоколов и прослушивающие программы («снифферы»);
- технические средства съема информации.

Ошибки в программном обеспечении. Самое узкое место любой сети. Программное обеспечение серверов, рабочих станций, маршрутизаторов и т. д. написано людьми, следовательно, оно практически всегда содержит ошибки. Чем выше сложность подобного ПО, тем больше вероятность обнаружения в нем ошибок и уязвимостей. Большинство из них не представляет никакой опасности, некоторые же могут привести к трагическим последствиям, таким, как получение злоумышленником контроля над сервером, неработоспособность сервера, несанкционированное использование ресурсов (хранение ненужных данных на сервере, использование в качестве плацдарма для атаки и т.п.). Большинство таких уязвимостей устраняется с помощью пакетов обновлений, регулярно выпускаемых производителем ПО. Своевременная установка таких обновлений является необходимым условием безопасности сети.

DoS и DDoS-атаки. Denial Of Service (отказ в обслуживании) - особый тип атак, направленный на выведение сети или сервера из работоспособного состояния. При DoS-атаках могут использоваться ошибки в программном обеспечении или легитимные операции, но в больших масштабах (например, посылка огромного количества электронной почты). Новый тип атак DDoS (Distributed Denial Of Service) отличается от предыдущего наличием огромного количества компьютеров, расположенных в большой географической зоне. Такие атаки просто перегружают канал трафиком и мешают прохождению, а зачастую и полностью блокируют передачу по нему полезной информации. Особенно актуально это для компаний, занимающихся каким-либо online-бизнесом, например, торговлей через Internet.

Компьютерные вирусы, троянские кони. Вирусы - старая категория опасностей, которая в последнее время в чистом виде практически не встречается. В связи с активным применением сетевых технологий для передачи данных вирусы все более тесно интегрируются с троянскими компонентами и сетевыми червями. В настоящее время компьютерный вирус использует для своего распространения либо электронную почту, либо уязвимости в ПО. А часто и то, и другое. Теперь на первое место вместо деструктивных функций вышли функции удаленного управления, похищения информации и использования зараженной системы в качестве плацдарма для дальнейшего распространения. Все чаще зараженная машина становится активным участником DDoS-атак. Методов борьбы достаточно много, одним из них является все та же своевременная установка обновлений.

Анализаторы протоколов и «снифферы». В эту группу входят средства перехвата передаваемых по сети данных. Такие средства могут быть как аппаратными, так и программными. Обычно данные передаются по сети в открытом виде, что позволяет злоумышленнику внутри локальной сети перехватить их. Некоторые протоколы работы с сетью (POPS, FTP) не используют шифрование паролей, что позволяет злоумышленнику перехватить их и использовать самому. При передаче данных по глобальным сетям эта проблема встает наиболее остро. По возможности следует ограничить доступ к сети неавторизированным пользователям и случайным людям.

Технические средства съема информации. Сюда можно отнести такие средства, как клавиатурные жучки, различные мини-камеры, звукозаписывающие устройства и т.д. Данная группа используется в повседневной жизни намного реже вышеперечисленных, так как, кроме наличия спецтехники, требует доступа к сети и ее составляющим.

Человеческий фактор:

- уволенные или недовольные сотрудники;
- промышленный шпионаж;
- халатность;
- низкая квалификация.

Уволенные и недовольные сотрудники. Данная группа людей наиболее опасна, так как многие из работающих сотрудников могут иметь разрешенный доступ к конфиденциальной информации. Особенную группу составляют системные администраторы, зачастую недовольные своим материальным положением или несогласные с увольнением, они оставляют «черные ходы» для последующей возможности злонамеренного использования ресурсов, похищения конфиденциальной информации и т. д.

Промышленный шпионаж. Это самая сложная категория. Если ваши данные интересны кому-либо, то этот кто-то найдет способы достать их. Взлом хорошо защищенной сети — не самый простой вариант. Очень может статься, что уборщица «тетя Глаша», моющая под столом и ругающаяся на непонятный ящик с проводами, может оказаться хакером весьма высокого класса.

Халатность. Самая обширная категория злоупотреблений: начиная с не установленных вовремя обновлений, неизмененных настроек «по умолчанию»

и заканчивая несанкционированными модемами для выхода в Internet, - в результате чего злоумышленники получают открытый доступ в хорошо защищенную сеть.

Низкая квалификация. Часто низкая квалификация не позволяет пользователю понять, с чем он имеет дело; из-за этого даже хорошие программы защиты становятся настоящей морокой системного администратора, и он вынужден надеяться только на защиту периметра. Большинство пользователей не понимают реальной угрозы от запуска исполняемых файлов и скриптов и считают, что исполняемые файлы -только файлы с расширением «exe». Низкая квалификация не позволяет также определить, какая информация является действительно конфиденциальной, а какую можно разглашать. В крупных компаниях часто можно позвонить пользователю и, представившись администратором, узнать у него учетные данные для входа в сеть. Выход только один -обучение пользователей, создание соответствующих документов и повышение квалификации.

1.1 Система обнаружения вторжений

Система обнаружения вторжений (СОВ) — программное или аппаратное средство, предназначенное для выявления фактов неавторизованного доступа в компьютерную систему или сеть либо несанкционированного управления ими в основном через Интернет. Соответствующий английский термин - *Intrusion Detection System (IDS)*. Системы обнаружения вторжений обеспечивают дополнительный уровень защиты компьютерных систем.

Системы обнаружения вторжений используются для обнаружения некоторых типов вредоносной активности, которая может нарушить безопасность компьютерной системы. К такой активности относятся сетевые атаки против уязвимых сервисов, атаки, направленные на повышение привилегий, неавторизованный доступ к важным файлам, а также действия вредоносного программного обеспечения (компьютерных вирусов, троянов и червей)

Обычно архитектура СОВ включает:

- сенсорную подсистему, предназначенную для сбора событий, связанных с безопасностью защищаемой системы;
- подсистему анализа, предназначенную для выявления атак и подозрительных действий на основе данных сенсоров;
- хранилище, обеспечивающее накопление первичных событий и результатов анализа;
- консоль управления, позволяющая конфигурировать СОВ, наблюдать за состоянием защищаемой системы и СОВ, просматривать выявленные подсистемой анализа инциденты.

Существует несколько способов классифицировать СОВ в зависимости от типа и расположения сенсоров, а также методов, используемых подсистемой анализа для выявления подозрительной активности. Во многих простых СОВ все компоненты реализованы в виде одного модуля или устройства.

Виды систем обнаружения вторжений. В сетевой СОВ, сенсоры расположены на важных для наблюдения точках сети, часто в демилитаризованной зоне, или на границе сети. Сенсор перехватывает весь сетевой трафик и анализирует содержимое каждого пакета на наличие вредоносных компонентов. Протокольные СОВ используются для отслеживания трафика, нарушающего правила определенных протоколов либо синтаксис языка (например, SQL). В хостовых СОВ сенсор обычно является программным агентом, который ведет наблюдение за активностью хоста, на который установлен. Также существуют гибридные версии перечисленных видов СОВ.

Сетевая СОВ (Network-based IDS, NIDS) отслеживает вторжения, проверяя сетевой трафик и ведет наблюдение за несколькими хостами. Сетевая система обнаружения вторжений получает доступ к сетевому трафику, подключаясь к хабу или свитчу, настроенному на зеркалирование портов, либо сетевое TAP устройство. Примером сетевой СОВ является Snort.

Основанное на протоколе СОВ (Protocol-based IDS, PIDS) представляет собой систему (либо агента), которая отслеживает и анализирует коммуникационные протоколы со связанными системами или пользователями. Для веб-сервера подобная СОВ обычно ведет наблюдение за HTTP и HTTPS протоколами. При использовании HTTPS СОВ должна располагаться на таком интерфейсе, чтобы просматривать HTTPS пакеты еще до их шифрования и отправки в сеть.

Основанная на прикладных протоколах СОВ (Application Protocol-based IDS, APIDS) — это система (или агент), которая ведет наблюдение и анализ данных, передаваемых с использованием специфичных для определенных приложений протоколов. Например, на веб-сервере с SQL базой данных СОВ будет отслеживать содержимое SQL команд, передаваемых на сервер.

Узловая СОВ (Host-based IDS, HIDS) - система (или агент), расположенная на хосте, отслеживающая вторжения, используя анализ системных вызовов, логов приложений, модификаций файлов (исполняемых, файлов паролей, системных баз данных), состояния хоста и прочих источников. Примером является OSSEC.

Гибридная СОВ совмещает два и более подходов к разработке СОВ. Данные от агентов на хостах комбинируются с сетевой информацией для создания наиболее полного представления о безопасности сети. В качестве примера гибридной СОВ можно привести Prelude.

Пассивные и активные системы обнаружения вторжений. В пассивной СОВ при обнаружении нарушения безопасности, информация о нарушении записывается в лог приложения, а также сигналы опасности отправляются на консоль и/или администратору системы по определенному каналу связи. В активной системе, также известной как Система Предотвращения Вторжений (IPS — Intrusion Prevention system (англ.)), СОВ ведет ответные действия на нарушение, сбрасывая соединение или перенастраивая межсетевой экран для блокирования трафика от злоумышленника. Ответные действия могут проводиться автоматически либо по команде оператора.

Хотя и СОВ, и межсетевой экран относятся к средствам обеспечения информационной безопасности, межсетевой экран отличается тем, что ограничивает поступление на хост или подсеть определенных видов трафика для предотвращения вторжений и не отслеживает вторжения, происходящие внутри сети. СОВ, напротив, пропускает трафик, анализируя его и сигнализируя при обнаружении подозрительной активности. Обнаружение нарушения безопасности проводится обычно с использованием эвристических правил и анализа сигнатур известных компьютерных атак.

История разработок СОВ. Первая концепция СОВ появилась благодаря Джеймсу Андерсону и статье. В 1984 Фред Коэн (см. Обнаружение вторжений) сделал заявление о том, что каждое вторжение обнаружить невозможно и ресурсы, необходимые для обнаружения вторжений, будут расти вместе с степенью использования компьютерных технологий.

Дороти Деннинг, при содействии Питера Неймана, опубликовали модель СОВ в 1986, сформировавшую основу для большинства современных систем. Ее модель использовала статистические методы для обнаружения вторжений и называлась IDES (Intrusion detection expert system - экспертная система обнаружения вторжений). Система работала на рабочих станциях Sun и проверяла как сетевой трафик, так и данные пользовательских приложений.

IDES использовала два подхода к обнаружению вторжений: в ней использовалась экспертная система для определения известных видов вторжений и компонент обнаружения, основанный на статистических методах и профилях пользователей и систем охраняемой сети. Тереза Лунт предложила использовать искусственную нейронную сеть как третий компонент для повышения эффективности обнаружения. Вслед за IDES в 1993 вышла NIDES (Next-generation Intrusion Detection Expert System - экспертная система обнаружения вторжений нового поколения).

MIDAS (Multics intrusion detection and alerting system), экспертная система, использующая P-BEST и LISP, была разработана в 1988 году на основе работы Деннинга и Неймана. В этом же году была разработана система Haystack, основанная на статистических методах.

W&S (Wisdom & Sense - мудрость и чувство), основанный на статистических методах детектор аномалий, был разработан в 1989 году в Лос-Аламосской Национальной лаборатории.[7] W&S создавал правила на основе статистического анализа и затем использовал эти правила для обнаружения аномалий.

В 1990, в TIM (Time-based inductive machine) было реализовано обнаружение аномалий с использованием индуктивного обучения на основе последовательных паттернов пользователя на языке Common LISP. Программа была разработана для VAX 3500. Примерно в то же время был разработан NSM (Network Security Monitor - монитор сетевой безопасности), сравнивающий матрицы доступа для обнаружения аномалий на рабочих станциях Sun-3/50. В том же 1990 году был разработан ISOA (Information Security Officer’s Assistant), содержащий в себе множество стратегий обнаружения, включая статистику, проверку профиля и экспертную систему. ComputerWatch, разработанный в

AT&T Bell Labs, использовал статистические методы и правила для проверки данных и обнаружения вторжений.

Далее, в 1991, разработчики Университета Калифорнии разработали прототип распределенной системы DIDS (Distributed intrusion detection system), которая также являлась экспертной системой. Также в 1991 сотрудниками Национальной Лаборатории Встроенных Вычислительных Сетей (ICN) была разработана система NADIR (Network anomaly detection and intrusion reporter). На создание этой системы оказала большое влияние работа Деннинга и Люнт. NADIR использовала основанный на статистике детектор аномалий и экспертную систему.

В 1998 году Национальная лаборатория им. Лоуренса в Беркли представила Bro, использующий собственный язык правил для анализа данных libpcap. [14] NFR (Network Flight Recorder), разработанный в 1999, также работал на основе libpcap. В ноябре 1998 был разработан APE, снифер пакетов, тоже использующий libpcap. Спустя месяц APE был переименован в Snort.

В 2001 году была разработана система ADAM IDS (Audit data analysis and mining IDS). Система использовала данные tcpdump для создания правил.

Свободно распространяемые СОВ:

- Snort NIDS;
- Bro NIDS;
- Prelude Hybrid IDS;
- OSSEC HIDS;
- Samhain HIDS;
- Suricata;
- Open Source Tripwire.

Коммерческие СОВ:

- (fr) CATNET;
- Check Point IPS Blade;
- Check Point IPS;
- McAfee IPS;
- IBM ISS Proventia IPS.

Типы систем обнаружения вторжений

Существует классическое деление СОВ на:

- системы уровня сети, на которые отводится трафик с маршрутизатора (network-based);
- системы уровня хоста, которые обнаруживают изменения на отдельно взятой машине, например анализируя журналы или сетевую активность (host-based);
- системы, основанные на оценке уязвимостей (vulnerability-assessment).

Решение выносится на основе полученных данных.

Т.е. задачи системы состоят из:

- получения данных;
- интерпретации полученных данных;

- представления результата.

Соответственно, все системы возможно позиционировать по значениям следующих признаков:

- тип собираемых данных;
- метод получения данных;
- метод интерпретации данных;
- метод представления результата.

Внесистемной характеристикой я считаю тип реакции на результат:

- информационный;
- активный.

В первом случае, производится информирование заинтересованных лиц.
Во втором — активные действия, например блокирование диапазона адресов атакующего.

По этому признаку данные системы обычно искусственно разделяются на IDS и IPS.

Характеристика эта внесистемная, потому что предполагается разбиение системы на "разведывательную" и "силовую" части. И любая IDS может быть включена в состав IPS.

Классы типа собираемых данных

Следуя классическому делению, я введу два класса и добавлю третий, чтобы возможно было представить класс, как "координату в конфигурационном пространстве" (просто для того, чтобы позже все перестановки быстро получить):

- данные, собираемые об узле сети;
- данные, собираемые обо всей сети;
- гибридная система.

Типы данных, собираемые об узле сети - это данные, которые касаются только одного узла и частично тех, кто с ним взаимодействует. Анализ таких данных позволяет ответить на вопрос: "Идёт ли атака на данный хост?"

Как правило, удобнее собирать эти данные непосредственно на узле, но это не обязательно.

Например, сетевые сканеры могут получить список открытых портов на конкретном узле извне, не имея возможности запускать там код.

Этот класс включает данные следующих типов (к каждому из которых относятся конкретные собираемые показатели):

- сетевая активность узла;
- сетевые настройки узла;
- данные о файлах (списки и контрольные суммы, метаданные, действия с файлами и т.п..);
- данные о процессах.

При этом, узлы могут быть как рабочими станциями, не предполагающими их использования в качестве серверов, предоставляющих службы, так и серверами.

Разделяется хосты на типы, потому что нужно выделить отдельный случай: хост может быть специально сделан уязвимым, с целью изучения методов атак (в том числе до обучения нейронных сетей) и выявления атакующих узлов.

Возможно предполагать, что любое взаимодействие с данным узлом будет являться попыткой атаки.

Данные, собираемые о сети - это цельная картина сетевого взаимодействия.

Как правило, полные сетевые данные не собираются, т.к. это ресурсоёмко и считается, что нарушитель либо не может находиться внутри сети, либо ему обязательно нужна связь с внешним миром (что, конечно, не обязательно так, ведь провести атаку возможно и на физически изолированную сеть, для чего существуют техники "преодоления воздушного зазора").

В этом случае, IDS анализирует трафик, идущий через маршрутизатор, для чего в маршрутизаторе имеется SPAN порт, с которого трафик перенаправляется в IDS.

В принципе, ничего не мешает собирать данные с узла, на котором работает IDS, это даже полезно и привносит дополнительный контроль.

Также возможно собирать трафик сети на её узлах. Но это заставляет сетевой адаптер узла работать в режиме захвата всего трафика, чего обычно не предполагается при штатной работе, плюс это явно избыточно (единственный вариант, когда это может быть полезно, с большой натяжкой, - распределённый анализ потока).

Классы методов получения данных:

- пассивный. система не влияет напрямую на работоспособность сети. она просто производит анализ трафика;
- активный. система пытается провести "разведку боем", активно воздействуя на сеть, например с целью найти знакомые сигнатуры в ответном трафике;
- смешанный. используются оба вышеуказанных класса.

Пассивное обнаружение. При пассивном обнаружении, система просто наблюдает за обстановкой. Большинство IDS используют данный класс методов. Системы уровня хоста также, обычно используют данный класс методов. Например, они не пытаются удалить системный файл из под пользователя и проверить, что он был удалён, а просто оценивают соответствие прав на данный файл, шаблону в базе, и если соответствие не наблюдается, выдают предупреждение.

Активный поиск уязвимостей. В данном классе методов ошибки провоцируются путём некоторых действий, как известных, так и неизвестных (fuzzy системы).

После этого, по базе анализируется реакция на данные действия. Данный класс методов характерен для сканеров уязвимостей.

Для интерпретации результатов применимы оба класса методов:

- есть вариант анализа ответов по базе (например, характерные ответы на типичные шаблоны sql инъекций), либо поведенческий анализ (как себя стала после этого цель вести и как отвечает на следующие запросы). например, посылка некорректно составленного ip пакета должна привести к падению уязвимого сервера, после чего он перестанет отвечать;

- есть вариант обнаружения аномальной активности. например, если после посылки icmp пакета размером 777 байт с заполнением 0xdeadbeef, в сети резко возрастает уровень сетевой активности, а затем падает - это аномалия (нормальная ситуация - уровень сетевой активности не меняется).

Плюсы очевидны:

- превентивное обнаружение вторжений;

- какая-либо атака может ни разу не произойти, но сеть будет оставаться к ней уязвимой. данный класс методов выявит потенциальные уязвимости;

- проверка сети аналогично атакующему, что повышает шансы обнаружить уязвимости.

Минусы:

- дополнительная нагрузка на сеть;

- возможность реализовать успешную атаку в процессе сканирования, например dos некоторых сервисов;

- как правило, зависимость от базы атак, которая устаревает;

- неочевидный минус — иллюзия безопасности. может возникнуть ощущение, что "если сканер ничего не нашёл, то всё нормально".

При этом, риски направленных атак всё-равно остаются. Этот риск, возможно в меньшей степени, имеет место быть и для других классов методов.

Классы методов интерпретации данных. Возможно выделить следующие классы, каждый из которых может включать несколько методов:

- методы обнаружения известных нарушений;

- методы обнаружения аномалий;

- смешанный метод, включающий оба вышеназванных.

Обнаружение известных нарушений. Сводится к поиску признаков уже известных атак.

Преимущества:

- методы почти не подвержены ложным срабатываниям;

- методы, реализованные как сопоставление с образцом, как правило выполняются достаточно быстро и не требуют больших затрат ресурсов.

Недостатком этих методов является невозможность обнаруживать неизвестные системе атаки.

В классическом варианте реализация предполагает сравнение сигнатур пакетов с сигнатурами в базе. Сравнение может быть, как точное, так и с шаблоном или по регулярному выражению.

Большинство известных IDS используют методы данного класса.

Сюда же возможно отнести методы нечёткого сравнения:

- обученный перцептрон;
- методы сравнения с нечёткой логикой.

Обнаружение аномалий

Суть в том, чтобы записать шаблон нормальной активности сети и реагировать на отклонения от этого шаблона.

Возможен вариант реализации в виде базы сигнатур "нормальных" пакетов в сети и статистической системы обнаружения отклонений, когда анализатор ищет какие-то редкие действия, или активности. События в системе исследуются статистически, чтобы найти те из них, проявление которых выглядит аномальным.

Представленный ниже детектор будет пытаться обнаружить аномалии.

Методы представления результата

Как показал анализ существующих решений, обычно используются два типа методов:

- двуклассовый. система отвечает на вопрос "есть ли проблемы?", ответом "да/нет". данный способ характерен для академических статей и исследований, но он может быть применим для написания модулей или датчиков реальных сов;
- многоклассовый. система отвечает на вопрос "какие сейчас есть проблемы?". если проблем нет, то "никакие".

Результат также может быть представлен с определённым уровнем достоверности. И любая нейросеть представляет результат, не как "да" или "нет", а как множество вероятностей результатов.

В существующих подходах вероятность аппроксимируется к единичной или нулевой и нигде дальше не учитывается.

Например, изо всех возможных атак на выходе сети, выбирается первая с самой высокой вероятностью

Типы IDS. На данном этапе параметр "Метод представления результата" не представляет интереса (он был введён для разделения детекторов).

Потому возможно выделить 26 типов систем.

Далее, я сведу их в меньшее количество типов:

- типы: 0, 1, 2; ids уровня узла, основанная на поиске сигнатур, обнаруживающая аномальное поведение, либо гибридная; подобных ids существует достаточно много; сюда же возможно отнести антивирусы;
- типы: 3, 4, 5; сканер уязвимостей уровня узла, основанный на поиске сигнатур, обнаруживающий аномальное поведение, либо гибридный; сканеры незакрытых уязвимостей; функционал встроен в антивирусы; с натяжкой в качестве представителя группы возможно выделить avz; далее я этот тип отдельно не разбираю;
- типы: 6, 7, 8; гибридные ids уровня узла, как отслеживающие атаки пассивно, так и позволяющие провести активное сканирование;
- типы: 9, 10, 11; ids уровня сети, основанная на поиске сигнатур, обнаруживающая аномальное поведение, либо гибридная;

- типы: 12, 13, 14; сканер уязвимостей уровня сети, основанный на поиске сигнатур, обнаруживающий аномальные реакции, либо гибридный;
- типы: 15, 16, 17; гибридная ids уровня сети, обнаруживающая аномальное поведение, либо гибридная; должна позволять, как проводить сканирование, так и обнаруживать проблемы в пассивном режиме; это комплексные решения уровня сети; обычно, они не существуют в таком виде и сливаются со следующим типом;
- тип 26 (типы 18-25 обычно не имеют смысла по отдельности и сводятся к данному типу); гибридная система, основанная на агентах, каждый из которых может иметь один из вышеперечисленных типов;
- отдельным пунктом, который не попадает в классификацию, потому что не является ids, но имеет место быть как вспомогательная система, является honeypot.

Ниже указаны примеры существующих решений, попадающие в вышеназванные группы.

Существующие решения. Действующие системы.

Snort. Snort является классической IDS уровня сети и анализирует трафик на совпадение с базой правил (фактически с базой сигнатур). Т.е., данная система ищет известные нарушения. Для Snort возможно несложно реализовать свой модуль, что и было сделано в одной из работ. На базе Snort реализовано много известных коммерческих решений, в том числе русских.

Помимо работы с базой сигнатур, построенная на базе Snort IDS, вполне может иметь в своём составе эвристические, нейросетевые и подобные модули обнаружения. Как минимум, существует в рабочем виде статистический детектор аномалий для Snort.

Suricata. Suricata также, как и Snort является системой уровня сети. У данной системы есть несколько особенностей:

- базы сигнатур совместимы со snort;
- оценивает не только сетевой/транспортный уровень, но работает и на уровне прикладных протоколов;
- есть возможность реализовывать правила на lua, интерпретируемом языке, что расширяет диапазон возможностей;
- может анализировать трафик между двумя хостами, в целом, а не только отдельные пакеты/соединения; это позволяет, например, обнаруживать попытки подбора паролей;
- есть подсистема ip reputation, позволяющая присваивать "уровень репутации" каждому ip адресу.

Т.е., эта система, хотя и обнаруживают известные нарушения, также как предыдущая, обладает большей адаптивностью и возможностью обучаться (уровень репутации хоста может изменяться в процессе работы системы и влиять на принятие ей решения).

Bro. Платформа для создания IDS уровня сети. Является гибридной системой, с упором на обнаружение известных нарушений. Работает на транспортном, сетевом уровне и уровне приложений. Поддерживает свой язык сценариев.

Имеется возможность обнаружения аномалий, например, множественное подключение к сервисам на разных портах — не свойственное для нормального узла поведение, которое будет обнаружено.

Это реализовано, во-первых, на основе на основе проверок передаваемых данных на нормальность (например, TCP-пакет со всеми установленными флагами, наверное тут что-то не то, несмотря на то, что он корректен).

Во-вторых, на базе политик, описывающих как должна функционировать сеть в норме.

Bro не только обнаруживает атаки, но также помогает при диагностике сетевых проблем (заявленный функционал).

Технически Bro реализован достаточно интересно: он не производит анализ трафика напрямую по признакам, а прогоняет пакеты на "событийной машине", которая преобразует поток пакетов в серию высокоуровневых событий.

Эту машину возможно считать дополнительным уровнем абстракции, позволяющим отобразить сетевую (как правило, хотя и не обязательно) активность в терминах нейтральных относительно политик (события просто сигнализируют, о том, что нечто произошло, но о том почему, событийный движок ничего не говорит, т.к. это задача интерпретатора политик).

Например, HTTP запрос будет преобразован в событие http_request с соответствующими параметрами, которое передаётся на уровень анализа.

Интерпретатор политик выполняет сценарий, в котором устанавливаются обработчики событий. В данных обработчиках могут рассчитываться статистические параметры трафика. При этом, обработчики могут сохранять контекст, а не просто реагировать на отдельно взятый пакет. Т.е., учитывается временная динамика, "история" потока.

Краткая схема работы ядра Bro показана на рисунке 1.2.

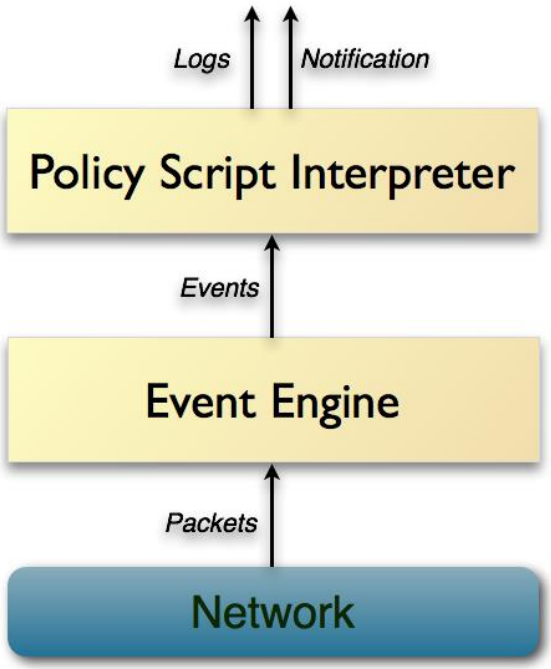

Рисунок 1.2 – система Bro

TripWire, OSSEC, Samhain. По классической терминологии, это представители систем уровня хоста. Совмещают в себе методы обнаружения известных нарушений и методы поиска аномальной активности.

Механизм поиска аномалий основан на том, что при установке, система сохраняет в базе хэши системных файлов и метаинформацию о них. При обновлении пакетов операционной системы, хэши пересчитываются.

В случае неконтролируемого изменения какого-либо наблюдаемого файла, через некоторое время система даст об этом знать (как правило, сканирование запускается планировщиком, хотя возможны системы, реагирующие на события ФС).

Помимо наблюдения за файлами, данные системы могут наблюдать за процессами, соединениями и анализировать системные журналы.

Также, ими может использоваться база известных нарушений, которая периодически обновляется.

Некоторые имеют централизованный интерфейс, который позволяет анализировать данные одновременно на многих узлах.

Пример развёртывания OSSEC приведен в соотстветсвии с рисунком 1.3.

Рисунок 1.3 – система TripWire, OSSEC, Samhain

Prelude SIEM, OSSIM. Это гибридные системы, позиционируемые, как SIEM. Prelude сочетает в себе сенсорную сеть и анализатор. Заявлено, что

данная система обеспечивает повышенный уровень безопасности, поскольку взломщик может обойти одну IDS, но сложность обхода многочисленных механизмов защиты возрастает экспоненциально.

Система хорошо масштабируется: по заявлениям разработчиков, сенсорная сеть может покрывать континент или даже мир.

Система совместима со многими существующими IDS по формату данных (IDMEF) в том числе с: AuditD, Nepenthes, NuFW, OSSEC, Pam, Samhain, Sancp, Snort, Suricata, Kismet и т.д..

Примерно тоже самое возможно сказать про OSSIM. Сканеры уязвимостей. Это системы, которые занимаются активным поиском уязвимостей на узле, либо в сети.

Самые простые ищут только известные проблемы, которые есть в базе, более серьёзные системы могут совмещать в себе как методы обнаружения известных нарушений, так и аномальной активности.

Сканеры уязвимостей появились давно и наделать их успели достаточно много.

Приведу только несколько известных продуктов, которые представлены на рынке:

- Nmap - бесплатный сетевой сканер, который используют скорее, как вспомогательный инструмент: он позволяет определить ОС, получить список открытых портов и сервисов, которые их открыли, и т.п… При этом, включает Netmap Scripting Engine, что позволяет на его основе строить автоматизированные решения (например, совместить сканирование с запуском брутфосера на нужных портах);
- XSpider - классический сканер уязвимостей с базой, эвристиками, поиском аномального поведения и т.д.;
- Metasploit — фреймворк, содержащий наборы эксплоитов, сканеры, готовые нагрузки, запускаемые на атакуемом узле и т.п… Позволяет провести исследование узла или группы узлов на подверженность известным уязвимостям;
- OpenVAS - фреймворк, сочетающий в себе сканер и решение для управления уязвимостями. Содержит базу с более чем 50000 уязвимостями, опознаёт указанные цели и проверяет их реакцию на применение соответствующих эксплоитов.

Возможности данных систем на примере вышеприведённого XSpider:

- идентификация сервисов на случайных портах для выявления уязвимостей серверов с нестандартной конфигурацией;
- эвристический метод определения типов и имен сервисов (http, ftp, smtp, pop3, dns, ssh и др.) для определения настоящего имени сервера и корректной работы проверок;
- обработка rpc-сервисов (windows и *nix) с полной идентификацией, включая определение детальной конфигурации компьютера;
- проверка слабости парольной защиты: оптимизированный подбор паролей практически во всех сервисах, требующих аутентификации;

- глубокий анализ контента веб-сайтов, включая выявление уязвимостей в скриптах: sqli, xss, запуск произвольных программ и др;
- анализ структуры http-серверов для поиск слабых мест в конфигурации;
- расширенная проверка узлов под управлением windows;
- проведение проверок на нестандартные dos-атаки;
- возможность автоматизации.

Honeypot

Системы специально сделанные уязвимыми, с целью привлечения к ним атак. Могут быть использованы для исследования действий атакующего.

Пример развёртывания приведен в рисунке 1.4:

- HoneyWeb - honeypot, запускающий WEB сервисы;
- Honeyd - позволяет создавать виртуальные хосты, на которых работают сервисы. Может эмулировать разные ОС.

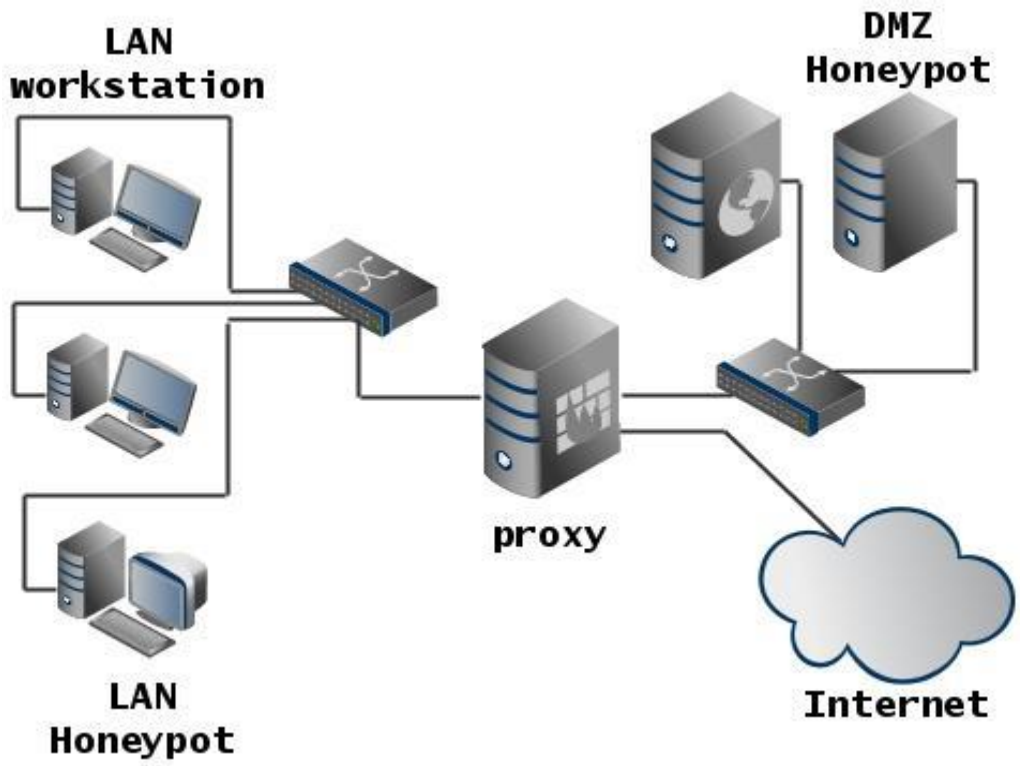

Рисунок 1.4 – Система Honeypot

Вообще, готовых honeypot решений достаточно много.

Сюда же могут относиться ловушки типа tarpits, замедляющие атаку (повышает требования к ресурсам атакующего, даёт время на реагирование), которые возможно считать "активным" вариантом honeypot.

1.2 Система предотвращения вторжений

Сетевая система обнаружения вторжений может защитить от атак, которые проходят через межсетевой экран во внутреннюю ЛВС.

Межсетевые экраны могут быть неправильно сконфигурированы, пропуская в сеть нежелательный трафик. Даже при правильной работе межсетевые экраны обычно пропускают внутрь трафик некоторых приложений, который может быть опасным. Порты часто переправляются с межсетевого экрана внутренним серверам с трафиком, предназначенным для почтового или другого общедоступного сервера. Сетевая система обнаружения вторжений может отслеживать этот трафик и сигнализировать о потенциально опасных пакетах.

Правильно сконфигурированная сетевая система обнаружения вторжений может перепроверять правила межсетевого экрана и предоставлять дополнительную защиту для серверов приложений.

Сетевые системы обнаружения вторжений полезны при защите от внешних атак, однако одним из их главных достоинств является способность выявлять внутренние атаки и подозрительную активность пользователей. Межсетевой экран защитит от многих внешних атак, но, когда атакующий находится в локальной сети, межсетевой экран вряд ли сможет помочь. Он видит только тот трафик, что проходит через него, и обычно слеп по отношению к активности в локальной сети. Считайте сетевую систему обнаружения вторжений и межсетевой экран взаимодополняющими устройствами безопасности - вроде надежного дверного замка и системы сигнализации сетевой безопасности. Одно из них защищает вашу внешнюю границу, другое -внутреннюю часть (рисунок 1.5).

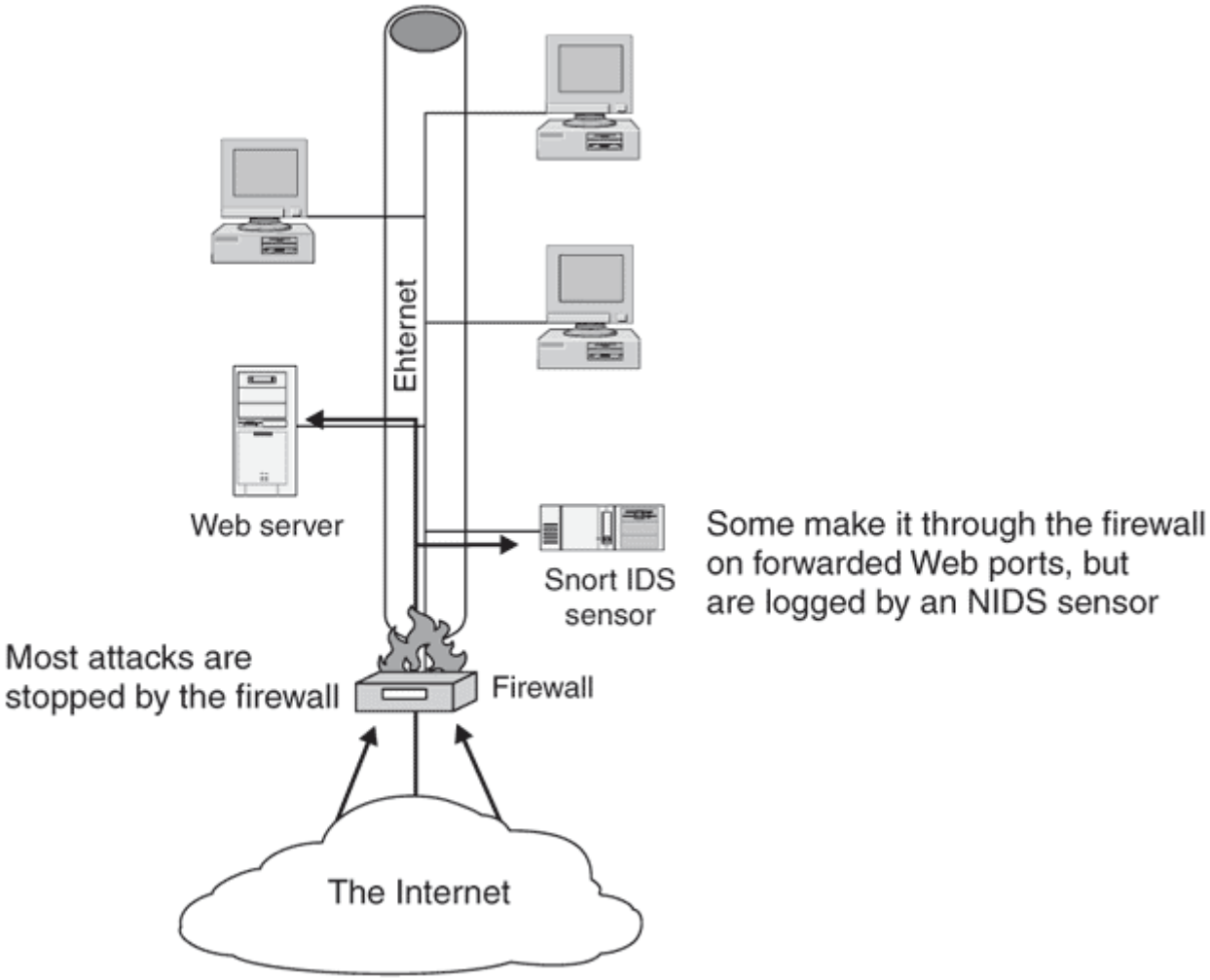

Рисунок 1.5 – Защита с помощью сетевой системы обнаружения вторжений и межсетевого экрана

Имеется веская причина, чтобы внимательно следить за трафиком внутренней сети. Как показывает статистика ФБР, более 70 процентов компьютерных преступлений исходят из внутреннего источника. Хотя мы склонны считать, что наши коллеги не сделают ничего, чтобы нам навредить, но иногда это бывает не так. Внутренние злоумышленники - не всегда ночные хакеры. Это могут быть и обиженные системные администраторы, и неосторожные служащие. Простое действие по загрузке файла или по открытию файла, присоединенного к электронному сообщению, может внедрить в вашу систему "троянскую" программу, которая создаст дыру в межсетевом экране для всевозможных бед. С помощью сетевой системы обнаружения вторжений вы сможете пресечь подобную активность, а также другие возможные компьютерные интриги. Хорошо настроенная сетевая система обнаружения вторжений может играть роль электронной "системы сигнализации" для вашей сети.

Новое поколение систем обнаружения вторжений. Системы обнаружения вторжений на основе выявления аномальной активности

Вместо применения статических сигнатур, с помощью которых можно выявлять только явно вредоносную деятельность, системы нового поколения

отслеживают нормальные уровни для различных видов активности в сети. Если наблюдается внезапный всплеск трафика FTP, то система предупредит об этом. Проблема с системами такого рода состоит в том, что они весьма склонны к ложным срабатываниям - то есть выдаче сигналов тревоги, когда в сети имеет место нормальная, допустимая деятельность. Так, в примере с FTP-трафиком загрузка особенно большого файла будет возбуждать сигнал тревоги.

Следует учитывать также, что системе обнаружения вторжений на основе выявления аномальной активности требуется время, чтобы построить точную модель сети. Вначале система генерирует так много тревожных сигналов, что пользы от нее почти никакой. Кроме того, подобные системы обнаружения вторжений можно обмануть, хорошо зная сеть. Если хакеры достаточно незаметны и используют протоколы, которые активно применяются в сети, они не привлекут внимания систем такого рода. С другой стороны, важное преимущество подобных систем - отсутствие необходимости постоянно обновлять набор сигнатур. Когда эта технология достигнет зрелости и достаточной интеллектуальности, она, вероятно, станет употребительным методом обнаружения вторжений.

Системы предотвращения вторжений. Новый тип сетевых систем обнаружения вторжений, называемый системами предотвращения вторжений, декларирован как решение всех проблем корпоративной безопасности. Основная идея состоит в том, чтобы при генерации тревожных сигналов предпринимать ответные действия, такие как написание на лету индивидуальных правил для межсетевых экранов и маршрутизаторов, блокирующих активность подозрительных IP-адресов, запрос или даже контратака систем-нарушителей.

Хотя эта новая технология постоянно развивается и совершенствуется, ей еще слишком далеко до проведения анализа и принятия решений на уровне человека. Факт остается фактом - любая система, которая на 100% зависит от машины и программного обеспечения, всегда может быть обманута посвятившим себя этому человеком (хотя некоторые проигравшие шахматные гроссмейстеры могут с этим не согласиться). Примером системы предотвращения вторжений с открытыми исходными текстами служит Inline Snort Джеда Хейла - свободный модуль для сетевой системы обнаружения вторжений Snort.

Проблема ложных срабатываний сетевых систем обнаружения вторжений

Одной из главных проблем систем обнаружения вторжений является их склонность к большому числу ложных срабатываний. Ложное срабатывание имеет место, когда система генерирует сигнал тревоги на основе того, что она считает вредоносной или подозрительной активностью, но что в действительности оказывается нормальным трафиком для данной сети. Обычно в подразумеваемой конфигурации сетевая система обнаружения вторжений будет реагировать на все хоть чуть-чуть необычное. У большинства подобных систем имеются обширные используемые по

умолчанию базы данных из тысяч сигнатур возможной подозрительной активности. Производители сетевых систем обнаружения вторжений не могут знать характер вашего сетевого трафика, поэтому для перестраховки они предусматривают срабатывание по каждому поводу.

Типичные причины ложных срабатываний. Работа системы мониторинга сети.

Многие организации используют системы мониторинга сети, такие как HP OpenView или WhatsUp Gold, чтобы следить за системами в своей сети. Они характеризуются высокой сетевой активностью опроса и обнаружения. Для опроса состояния эти системы обычно применяют SNMP или аналогичный протокол, но они могут также использовать эхо-тестирование и другие, более назойливые проверки. По умолчанию большинство систем обнаружения вторжений рассматривают эту активность как вредоносную или, по крайней мере, подозрительную. В большой сети мониторинг может порождать тысячи сигналов тревоги в час, если система обнаружения вторжений настроена для отслеживания такой деятельности. Этого можно избежать, игнорируя активность с участием IP-адреса системы мониторинга. Можно также исключить из базы данных соответствующие сигналы тревоги, если их отслеживание не представляет для вас особой важности.

Сетевое сканирование уязвимостей/сканеры портов. Всякий раз, когда вы запускаете сетевое тестирование уязвимостей или сканирование портов с помощью таких программ, как Nessus и Nmap, ваша сетевая система обнаружения вторжений будет сходить с ума. Эти программы созданы для выполнения именно того, что делают хакеры. На самом деле, вероятно, сигналы тревоги заданы для большинства встраиваемых модулей Nessus. И здесь также можно отключить сообщения с участием IP-адреса сервера Nessus или Nmap, но лучше всего вообще выключать систему обнаружения вторжений на время планового сканирования. В этом случае сканирующая машина будет по-прежнему защищена от атак, когда не выполняется сканирование, а база данных сигналов тревоги не будет искажена множеством данных от вашей собственной активности по сканированию.

Пользовательская активность. Большинство сетевых систем обнаружения вторжений настроены для сигнализации об опасной активности пользователей, такой как одноранговое разделение файлов, мгновенный обмен сообщениями и т.д. Однако, если подобная активность допускается либо формальной политикой, либо просто несоблюдением существующих политик, то она будет фиксироваться в журналах в виде сигналов. Это может стать основанием для проведения в жизнь или создания политик против таких видов деятельности, так как можно показать, какую часть полосы пропускания и сколько времени они занимают, не говоря уже о последствиях для безопасности. Однако, если вы намерены и далее разрешать такую активность, то необходимо закомментировать эти правила, чтобы не заполнять журналы сигналами ложных срабатываний.

Поведение, напоминающее "троянскую" программу или "червя".

Современные вирусы и вирусоподобное программное обеспечение ("черви" и "троянские" программы) нередко используют сетевые средства, пытаясь выполнять такие действия, как инфицирование других машин или массовая рассылка электронных сообщений. Подобную активность можно выявить с помощью сетевых систем обнаружения вторжений. Однако эти сигнатуры могут порождать сигналы тревоги и при нормальной деятельности. Примером служит червь Nimda, который пытается копировать на различные системы файлы с определенными расширениями, такими как .eml. К сожалению, программа Microsoft Exchange ведет себя аналогично при использовании ее web-интерфейса. Поэтому, хотя знать о подобной "троянской" активности в сети было бы полезно, можно при желании отключить сигналы, порождаемые известной нормальной деятельностью, даже когда имеется потенциальная опасность, что трафик все-таки окажется вредоносным. Это поможет избежать чрезмерного количества ложных срабатываний.

Длинные базовые цепочки аутентификации.

Сигнал такого типа ориентирован на чрезмерно длинные входные строки Web, поскольку некоторые программы использования уязвимостей применяют подобный метод для переполнения буфера и несанкционированного получения доступа. Однако в последнее время многие Web-сайты набивают в это поле много информации и могут ненароком сбить с толку сетевую систему обнаружения вторжений.

Аутентификационная активность базы данных.

Некоторые сетевые системы обнаружения вторжений следят за деятельностью по администрированию баз данных. Теоретически в производственных базах данных не должно наблюдаться высокой административной активности, а ее наличие может служить признаком того, что кто-то пытается что-то сделать с базой. Однако во многих базах данных использование идет параллельно с разработкой, отсюда и большой объем администрирования. Эта деятельность, хотя и вполне законная, будет порождать множество сигналов тревоги. Если ваша база данных находится в состоянии непрерывного развития, то вам, вероятно, следует отключить эти сигналы, по крайней мере пока база не стабилизируется и не перейдет в режим производственной эксплуатации.

Существует много других причин ложных срабатываний, зависящих от конфигурации сети и уровня активности. В подразумеваемой конфигурации сетевая система обнаружения вторжений может порождать сотни ложных срабатываний в день, что способно привести системного администратора в отчаяние. В результате сигналы тревоги этих систем вскоре начинают игнорироваться, как некий посторонний шум. Однако при небольших усилиях и с помощью методов сетевая система обнаружения вторжений может быстро стать полезным средством, а не электронной версией мальчика, который то и дело кричал "Волк!".

Как получить максимум пользы от системы обнаружения вторжений.

Чтобы реализовать истинный потенциал системы обнаружения вторжений, необходимо сделать несколько вещей, как перед установкой, так и после нее.

Правильное конфигурирование системы.

Если вы только что установили систему обнаружения вторжений и запустили ее в подразумеваемой конфигурации, то тысячи ложных срабатываний скоро переполнят чашу вашего терпения. Хотя вы сможете перенастроить систему постфактум, лучше поберечь силы и нервы, уделив некоторое время упреждающему конфигурированию. Не принимайте подразумеваемые настройки, индивидуализируйте их для своей ЛВС.

Большинство систем обнаружения вторжений группирует сигналы тревоги по категориям. Просмотрите каждую группу, чтобы решить, насколько она подходит для вашей сети. Если имеется группа сигнатур для UNIX-платформ, но у вас в сети нет UNIX-систем, то, вероятно, можно безопасно отключить весь этот пакет сигналов. В некоторых системах предусмотрены сигналы тревоги, зависящие от политики и отвечающие за такие вещи, как использование мгновенного обмена сообщениями или программного обеспечения одноранговых сетей. Если у вас уже есть системы, фильтрующие подобные виды активности, или вы их разрешаете, то эти сигналы можно отключить. Необходимо тщательно проанализировать группы сигналов. Хотя вам может пригодиться большинство сигналов для Windows-платформ, некоторые из них, возможно, не имеют отношения к вашей сети или будут вызывать ложные срабатывания.

Можно также освободить некоторые хосты от контроля. Если ваша персональная машина постоянно посылает в сеть SNMP-запросы, или вы постоянно входите как администратор, то это может порождать много бесполезных сигналов тревоги. Хотя освобождение от контроля снижает уровень безопасности и может оставить критически важные машины без защиты, оно способно сделать систему обнаружения вторжений более эффективной. Уделив несколько часов тщательному конфигурированию системы до ее активации, можно сберечь много времени и сил в будущем.

Настройка системы обнаружения вторжений.

Когда система запущена и работает, даже при скрупулезном конфигурировании она начнет генерировать сигналы тревоги. Вскоре, если вы найдете время для их анализа и начнете деактивировать правила, которые не подходят для вашей сети, вы сможете снизить число ложных срабатываний. Попутно у вас появится понимание того, как работает ваша сеть и какие потоки данных текут по ней, что полезно для любого сетевого администратора. Каждую неделю выделяйте некоторое время для модификации настроек системы обнаружения вторжений. Некоторые системы позволяют относительно легко пометить сигнал как источник ложных срабатываний, тогда как другие заставят вас преодолеть некоторые препятствия. В среднем требуется несколько месяцев, чтобы настроить систему обнаружения вторжений на выдачу полезных сигналов о наказуемой

активности, и то только в случае целенаправленных усилий по тонкой настройке.

Средства анализа для систем обнаружения вторжений.

Системы обнаружения вторжений обычно предлагают администраторам несколько различных способов получения уведомлений о срабатывании сигналов тревоги. В простейшем случае сигналы могут просто протоколироваться для последующего просмотра. На самом деле это не рекомендуется, так как заставляет администратора неусыпно следить за регистрационными журналами. Если не делать этого ежедневно, то могут пройти дни или недели, прежде чем попытки вторжения будут обнаружены. Другой возможностью извещения соответствующего должностного лица о возникновении сигнала тревоги является отправка сообщения по электронной почте или на пейджер. Однако даже с хорошо настроенной системой получение на пейджер по несколько сообщений в день может доставлять слишком много хлопот. Кроме того, электронные сообщения будут иметь формат, в котором их сложно сравнивать с прошлыми сигналами тревоги или анализировать каким-то иным образом. Лучшим способом обработки сигналов тревоги является их немедленное занесение в базу данных, чтобы можно было выполнить углубленный анализ. Существует средство с открытыми исходными текстами для систем обнаружения вторжений, называемое ACID (Analysis Console for Intrusion Detection - консоль анализа для обнаружения вторжений). Оно подробно рассматривается в "Средства анализа и управления" .

Система предотвращения вторжений будет работать путем сканирования всего сетевого трафика. Для этого инструмент IPS обычно располагается прямо за брандмауэром, выступая в качестве дополнительного уровня, который будет отслеживать события для вредоносного контента. Таким образом, инструменты IPS размещаются на прямых путях связи между системой и сетью, что позволяет инструменту анализировать сетевой трафик.

Ниже приведены три общих подхода для инструмента IPS для защиты сетей:

- обнаружение на основе сигнатур, при котором инструмент IPS использует предварительно определенные сигнатуры атак известных сетевых угроз для обнаружения угроз и принятия мер;
- обнаружение на основе аномалий, при котором IPS ищет неожиданное поведение сети и блокирует доступ к хосту в случае обнаружения аномалии;
- обнаружение на основе политик, при котором IPS сначала требует от администраторов создания политик безопасности - когда происходит событие, которое нарушает определенную политику безопасности, системным администраторам отправляется предупреждение.

Если обнаруживаются какие-либо угрозы, инструмент IPS обычно способен отправлять оповещения администратору, отбрасывать любые вредоносные сетевые пакеты и сбрасывать соединения путем перенастройки

брандмауэров, переупаковки полезных нагрузок и удаления зараженных вложений с серверов.

Инструменты IPS могут помочь отразить атаки типа «отказ в обслуживании» (DoS), распределенные атаки типа «отказ в обслуживании» (DDoS), черви, вирусы или эксплойты, такие как эксплойт нулевого дня. По словам Майкла Рида (Michael Reed), ранее принадлежавшего к сетям верхнего уровня (приобретенным Corero), эффективная система предотвращения вторжений должна выполнять более сложные мониторинг и анализ, такие как наблюдение и реагирование на шаблоны трафика, а также на отдельные пакеты. «Механизмы обнаружения могут включать в себя сопоставление адресов, сопоставление строк и подстрок HTTP [Протокол передачи гипертекста], сопоставление общих шаблонов, анализ соединения TCP [Протокол управления передачей], обнаружение аномалий пакетов, обнаружение аномалий трафика и сопоставление портов TCP / UDP [User Datagram Protocol] "

Типы систем предотвращения вторжений. Обычно появляются три типа систем предотвращения вторжений. Эти типы следующие:

- анализ поведения сети (NBA), который анализирует поведение сети на предмет ненормального потока трафика - обычно используется для обнаружения DDoS-атак;
- сетевая система предотвращения вторжений (NIPS), которая анализирует сеть на предмет выявления подозрительного трафика - как правило, окружающих протоколов;
- хост-система предотвращения вторжений (HIPS), которая устанавливается на одном хосте и используется для анализа подозрительной активности на одном конкретном хосте.

Кроме того, существуют другие типы инструментов IPS, в том числе анализирующие беспроводные сети. В целом, однако, можно сказать, что система предотвращения вторжений включает в себя любой продукт или практику, используемую для предотвращения доступа злоумышленников к вашей сети, такую как брандмауэры и антивирусное программное обеспечение.

Преимущества систем предотвращения вторжений включают следующее:

- снижение шансов на инциденты безопасности;
- обеспечение динамической защиты от угроз;
- автоматическое уведомление администраторов при обнаружении подозрительной активности;
- смягчение атак, таких как угрозы нулевого дня, DoS-атаки, DDoS-атаки и попытки атаки методом перебора;
- сокращение обслуживания сетей для ИТ-персонала;
- разрешить или запретить определенный входящий трафик в сеть.

Недостатки систем предотвращения вторжений включают следующее:

-когда система блокирует ненормальную активность в сети, предполагая, что она является вредоносной, это может быть ложным срабатыванием и привести к dos законному пользователю;

-если организации не хватает пропускной способности и пропускной способности сети, инструмент ips может замедлить работу системы;

-если в сети несколько ips, данные должны будут проходить через каждый из них, чтобы достичь конечного пользователя, что приведет к снижению производительности сети.

IPS также может быть дорогим.

IPS против IDS. IDS- это программные инструменты, предназначенные для обнаружения и мониторинга сетевого трафика. Инструменты IPS и IDS будут считывать сетевые пакеты и сравнивать их содержимое с известными угрозами. Однако IDS отличается тем, какие действия предпринимаются дальше. Инструмент IDS не будет предпринимать никаких действий самостоятельно. IDS требует, чтобы человек анализировал результаты и принимал решения о том, что делать дальше. Вот почему IPS рассматривается как расширение IDS.

IDS предназначен для мониторинга сети и отправки оповещений администраторам при обнаружении угрозы. Тем не менее, IPS предназначен для контроля доступа к сети и защиты сети от вреда.

Как и IDS, IPS будет контролировать сетевой трафик. Однако, поскольку эксплойт может быть выполнен быстро после того, как злоумышленник получит доступ, системы предотвращения вторжений также могут предпринять немедленные действия на основе набора правил, установленных администратором сети. Например, IPS может отбросить пакет, который он считает вредоносным, и заблокировать весь дальнейший трафик с этого IP-адреса или порта Интернет-протокола. Между тем, законный трафик должен быть направлен получателю без явного нарушения или задержки обслуживания.

2 ИСПОЛЬЗОВАНИЕ НЕЙРОННЫХ СЕТЕЙ В СОВ

2.1 Варианты подходов к решению задачи и нейросетевые решения

В работе проведён анализ упоминаний широко используемой базы KDD. И представлено следующее распределение методов.

Как видно, нейронные сети в работах за 2015-2016 годы представлены не столь широко, к тому же, это как правило, сети прямого распространения.

На практике же, решения, в том числе и приведённые выше, в основном опираются на следующие технологии анализа.

Методы обнаружения известных нарушений:

- сигнатурный анализ. сравнение сигнатуры данных, либо сигнатуры поведения с сигнатурой в обновляемой базе. сигнатура может быть представлена также шаблоном или регулярным выражением;
- экспертные системы на основе правил.

Методы обнаружения аномалий:

- пороговые детекторы (реагирующие, например, на стабильное превышение загрузки cpu на сервере) ;
- статистические системы (например, байесовские классификаторы, либо системы обучаемых классификаторов);
- поведенческий анализ (возможно отнести к нечётким правилам);
- использование модели "искусственная имунная система".

Про некоторые варианты решений, которых я не коснулся, возможно также посмотреть здесь.

У данных решений есть свои проблемы:

- сигнатурный поиск не реагирует на неизвестные атаки, и достаточно не сильно больших изменений сигнатуры атаки, для того, чтобы детектор её более не обнаружил;
- для экспертных систем, основанных на правилах, также как и для сигнатурного поиска, требуется поддерживать актуальность базы;
- для систем основанных на правилах, незначительные вариации в последовательности действий при атаке могут повлиять на процесс сравнения "деятельность-правило" до такой степени, что атака не обнаруживается соответствующим механизмом обнаружения. увеличение уровня отвлеченности у таких систем предоставляет частичное решение этой задачи, однако это сильно увеличивает число ложных срабатываний [10];
- системы на основе правил зачастую не обладают достаточной гибкостью в структуре правил;
- статистические системы не чувствительны к порядку следования событий (что верно не для всех существующих систем);
- для них и для пороговых детекторов трудно задать пороговые значения отслеживаемых системой обнаружения атак характеристик;
- с течением времени, статистические системы могут быть переобучены нарушителями так, чтобы атакующие действия рассматривались как нормальные.

Нейронные сети, прежде всего стоит рассматривать, в качестве замены статистических детекторов аномалий, либо дополнения к ним, но в какой-то мере, они могут заменить сигнатурный поиск и прочие методы.

Кроме того, гибридные системы, скорее всего будут применяться в ближайшем будущем.

Нейронные сети имеют, как свои преимущества:

- возможность анализа неполных входных данных или зашумлённого сигнала;
- отсутствие необходимости формализации знаний (заменяется обучением);
- отказоустойчивость: выход из строя некоторых элементов сети или разрушение связи не всегда делает сеть полностью неработоспособной;
- возможность простого распараллеливания работы;
- нейронные сети требуют меньшего вмешательства оператора;
- существует вероятность обнаружения неизвестных атак;
- сеть способна обучаться автоматически и в процессе работы;
- возможность обработки многомерных данных без сильного увеличения трудоемкости.

Возможно у вас возникло ложное ощущение, что нейронные сети вот уже сейчас заменяют компьютеры. Можно выбрасывать свои системные блоки и приобретать новомодные нейрокомпьютеры.

Однако это не так. У нейронных сетей есть ряд серьезных недостатков, которые тоже можно вывести из биологических нейронных сетей.

Недостатки:

- большинство подходов являются эвристическими и часто не приводят к однозначным решениям;
- для построения модели объекта на основе нейростей требуется предварительное обучение сети, что требует вычислительных и временных затрат;
- для того, чтобы обучить сеть, надо подготовить обучающую и тестовую выборки, что не всегда просто;
- обучение сети в ряде случаев приводит к тупиковым ситуациям: сети могут быть подвержены переобучению, либо не сходиться;
- поведение сети не всегда может быть однозначно предсказуемо, что вносит риски ложных срабатываний, либо пропуска инцидентов;
- трудно объяснить, почему сеть приняла то или иное решение (проблема вербализации);
- следовательно невозможно гарантировать повторяемость и однозначность получения результатов.

2.2 Существующие решения и предложения, основанные на нейронных сетях

Обобщу подходы, описанные в литературе:

- в работах Мустафаева [2], Жигулина и Подворчана [6], Halenar [7] предлагается использовать многослойный перцептрон, предварительно обученный на базе атак (например, KDD);
- в работе Haibo Zhang [3] и соавторов предлагается использовать нейронную сеть и вэйвлет-преобразование;
- Min-Joo Kang [4] и соавторы используют глубокое обучение для обнаружения проблем в бортовой CAN сети автомобиля;
- талалаев и соавторы [8] предлагают использовать рециркуляционную нейронную сеть или метод главных компонент для сжатия пространства признаков, после чего исследуют применение как двуслойного перцептрона, так и карты Кохонена;
- балахонцев и соавторы [10] используют трёхслойный перцептрон;
- Корнев и Пылькин [13] исследуют существующие подходы и указывают на возможность применения перцептронов с разным количеством слоёв, однослойных классификаторов для детектирования нормального состояния, а также гибридной сети, состоящей из карты Кохонена и перцептрона.

И практически используемые:

- компания DeepInstinct предлагает решения с использованием глубокого обучения, но деталей технологии, которую они используют, и каких-либо сравнений я не нашёл. Судя по тому, что у них представлено на сайте, они используют обучение с учителем;
- по неофициальным данным, разработка нейросетевых детекторов активно ведётся компаниями, в том числе и несколькими российскими.

Выводы по существующим решениям. Средства управления и мониторинга СПД эволюционируют в сторону комплексных решений.

Современные системы, в целом стремятся к тому, чтобы не только выполнять узкую задачу обнаружения вторжений, но и помогать в диагностике неисправностей сети, при этом реализуя как методы обнаружения аномалий, так и методы обнаружения известных нарушений.

В состав такой комплексной системы входят различные распределённые по сети сенсоры, которые могут быть, как пассивными, так и активными. Используя данные сенсоров, которыми могут являться целые экземпляры IDS, центральный коррелятор (в терминологии Prelude SIEM) анализирует общее состояние сети.

При этом, возрастают объёмы данных, которые требуется обработать, и их размерность.

В литературе, в основном, рассмотрены нейросетевые методы, основанные на перцептронах, либо картах Кохонена. Есть не заполненная ниша в виде разработок, на базе последних исследований, касающихся соревновательных нейросетей или свёрточных сетей. Хотя данные методы

хорошо себя зарекомендовали в смежных областях, где требуется сложный анализ.

Явно наблюдается разрыв между потребностями рынка и предложением в виде существующих решений.

Поэтому, имеет смысл проводить исследования в данном направлении.

2.3 Нейронный газ

Растущий нейронный газ. Итак, GNG – это алгоритм, позволяющий осуществлять адаптивную кластеризацию входных данных, т.е. не только разделить пространство на кластеры, но и определить необходимое их количество исходя из особенностей самих данных.

Начиная всего с двух нейронов, алгоритм последовательно изменяет (по большей части, увеличивает) их число, одновременно создавая набор связей между нейронами, наилучшим образом отвечающую распределению входных векторов, используя подход соревновательного хеббовского обучения. У каждого нейрона имеется внутренняя переменная, в которой накапливается т.н. «локальная ошибка». Соединения между узлами характеризуются переменной, называемой «возраст».

Псевдокод GNG выглядит следующим образом:

Инициализация: создать два узла с векторами весов, разрешенными распределением входных векторов, и нулевыми значениями локальных ошибок; соединить узлы связью, установив ее возраст равным 0.

Подать на вход нейросети вектор x.

Найти два нейрона s и t, ближайших к x, т.е. узлы с векторами весов w_s и w_t, такими, что $\|w_s - x\|^2$ минимальное, а $\|w_t - x\|^2$ второе минимальное значение расстояния среди всех узлов.

Обновить локальную ошибку нейрона-победителя s путем добавления к ней квадрата расстояния между векторами w_s и x:

$$E_s \leftarrow E_s + \|w_s - x\|^2 \tag{1}$$

Сместить нейрон-победитель s и всех его топологических соседей (т.е. все нейроны, имеющие соединение с победителем) в сторону входного вектора x на расстояния, равные долям ϵ_w и ϵ_n от полного.

$$w_s \leftarrow w_s + \epsilon_w * (w_s - x) \tag{2}$$

Увеличить на 1 возраст всех соединений, исходящих от победителя s.

Если два лучших нейрона s и t соединены, обнулить возраст их связи. В противном случае создать связь между ними.

Удалить все соединения, возраст которых превышает age_{max}. Если после этого имеются нейроны, не имеющие связей с другими узлами, удалить эти нейроны.

Если номер текущей итерации кратен λ, и предельный размер сети не достигнут, создать новый нейрон r по следующим правилам:

Найти нейрон u с наибольшей локальной ошибкой.

Среди соседей u найти нейрон v с максимальной ошибкой.

Создать узел r "посредине" между u и v:

$$w_r = \frac{w_u + w_v}{2} \tag{3}$$

Заменить связь между u и v на связи между u и r, v и r.

Уменьшить ошибки нейронов u и v, установить значение ошибки нейрона r.

$$\begin{gathered} E_u \leftarrow E_u * \alpha \\ E_v \leftarrow E_v * \alpha \\ E_r \leftarrow E_u \end{gathered} \tag{4}$$

Уменьшить ошибки всех нейронов j на долю β.

$$E_j \leftarrow E_j - E_j * \beta \tag{5}$$

Если критерий останова не выполнен, перейти к шагу 2.

Разберемся, каким же образом происходит адаптация растущего нейронного газа к особенностям входного пространства.

Первым делом, обратим внимание на увеличение переменной локальной ошибки у нейрона-победителя на шаге 4. Эта процедура приводит к тому, что наиболее часто выигрывающие узлы, т.е. те, в окрестности которых попадает наибольшее количество входных сигналов, имеют наибольшее значение ошибки, а значит, именно эти области становятся главными кандидатами на «уплотнение» путем добавления новых узлов.

Смещение узлов в сторону входного вектора на шаге 5 означает, что победитель стремится «усреднить» свое положение среди входных сигналов, расположенных в его окрестностях. При этом лучший нейрон немного «подтягивает» в сторону сигнала и своих соседей (как правило, выбирается $\epsilon_w \sim 100 \ldots 1000 \epsilon_n$).

Поясним работу со связями между нейронами на шагах 6-8. Не в даваясь в подробности, скажем, что смысл старения и удаления старых связей состоит в том, чтобы топология сети в целом максимально приближалась к так называемой триангуляции Делоне, т.е. такой триангуляции (разбиению на треугольники) множества нейронов, при которой, в частности, максимизируется минимальный угол треугольников разбиения (исключается появление «тонких» треугольников).

Попросту говоря, триангуляция Делоне соответствует наиболее «красивой» в смысле максимума энтропии топологизации слоя. Следует

отметить, что топологическая структура нужна нам не сама по себе, ценность представляет ее использование для определения местоположения новых узлов при их вставке на шаге 8 – они всегда располагаются посредине того или иного ребра.

9 шаг представляет собой коррекцию переменных ошибок всех нейронов слоя. Это необходимо для того, чтобы сеть «забывала» старые входные векторы и лучше реагировала на новые. Таким образом, достигается возможность использовать растущий нейронный газ для адаптации нейросети под нестационарные, а именно, медленно дрейфующие распределения входных сигналов. Это, однако, не дает ей способности отслеживать быстрые изменения в характеристиках входов (подробнее об этом ниже в разделе, где будут обсуждены недостатки алгоритма).

Пожалуй, отдельно стоит остановиться на критерии останова. Здесь алгоритм оставляет поле для фантазий разработчикам систем анализа. Возможные варианты – проверка эффективности сети на тестовом множестве, анализ динамики средней ошибки нейронов, ограничения на сложность самой сети и т.д.

Мы в ознакомительных целях ограничимся наиболее простым вариантом – поскольку целью данной статьи является демонстрация не столько алгоритма как такового, сколько возможности его реализации средствами MQL5, обучение слоя будем продолжать до тех пор, пока у нас попросту не закончатся входные сигналы (их количество, мы, естественно, зададим заранее).

Выбор способа организации данных. При программировании алгоритма нам, очевидно, придется столкнуться с необходимостью хранить в памяти то, что принято называть «множествами». Их у нас будет два – множество нейронов и множество связей между ними. Поскольку и та и другая структура будут видоизменяться в процессе работы программы (причем мы планируем как добавлять, так и удалять элементы), следует предусмотреть механизмы и для этого.

Конечно, можно было бы попытаться использовать динамические массивы объектов, однако при этом пришлось бы осуществлять множество операций копирования-перемещения данных, что существенно замедлило бы работу программы. Более подходящим вариантом для работы с абстракциями, обладающими указанными свойствами, являются программные графы, в частности, их простейший вариант – связный список.

Напомним читателям принцип работы связного списка (Рисунок 2.1). Объекты базового класса содержат в качестве одного из членов указатель на такой же объект, что позволяет объединять их в линейные структуры, независимые от физического порядка расположения объектов в памяти. В дополнение к этому имеется класс-«каретка», в который инкапсулируются процедуры перемещения по списку, добавления, вставки и удаления узлов, поиска, сравнения и сортировки, а также, при необходимости, и другие.

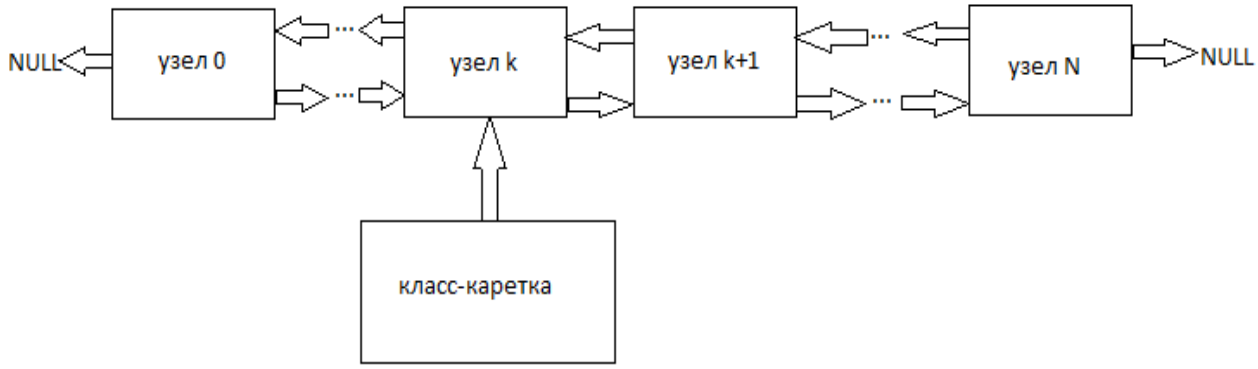

Рисунок 2.1 – Схематическое изображение организации линейных связных списков

Специалистами компании Metaquotes Software Corp. связные списки объектов класса CObject уже реализованы в стандартной библиотеке. Соответствующий программный код расположен в заголовочном файле List.mqh, который находится в папке MQL5\Include\Arrays стандартной поставки терминала MetaTrader 5.

Мы не будем изобретать велосипед и доверимся квалификации уважаемых программистов Metaquotes, взяв классы CObject и CList за основу наших структур данных. В этом нам поможет один из столпов объектно-ориентированного подхода – механизм наследования.

Известные проблемы. Как уже упоминалось, основной проблемой GNG является его неспособность отслеживать нестационарные ряды с быстро меняющимися характеристиками. Такие «прыгающие» распределения входных сигналов могут явиться причиной того, что значительная часть нейронов GNG-слоя, уже наработав определенную топологическую структуру, внезапно оказываются не у дел.

При этом, поскольку входные сигналы уже не попадают в область их расположения, возраст связей между этими нейронами не увеличивается, следовательно, «мертвая» часть сети, которая «помнит» старые характеристики сигнала, не делает полезной работы, а лишь расходует вычислительные ресурсы (см. Рисунок 2.2).

В случае медленно дрейфующих распределений такого негативного эффекта не наблюдается: если скорость дрейфа сравнима со «скоростью перемещения» нейронов при адаптации весов, GNG оказывается способным отслеживать такие изменения.

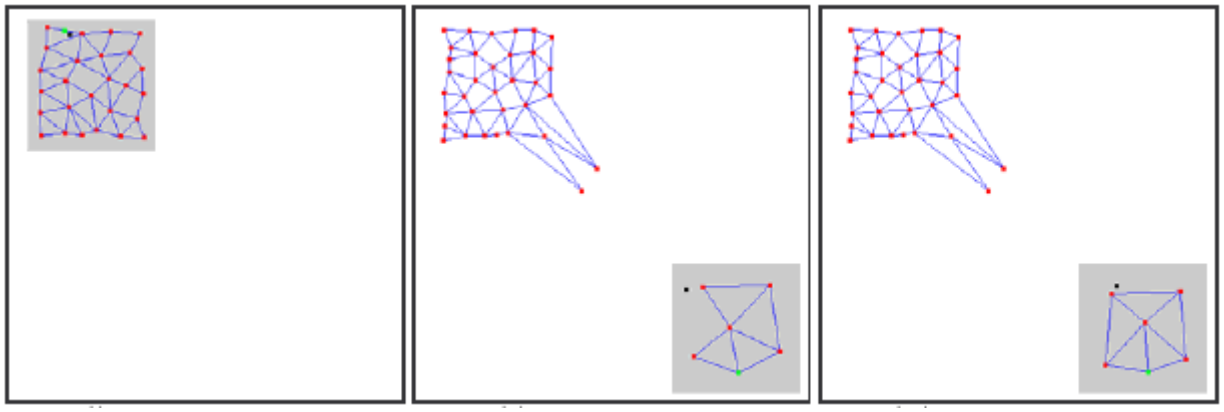

Рисунок 2.2 – Реакция растущего нейронного газа на "прыгающее" распределение

Отдельные неактивные (мертвые) узлы могут также появиться в сети, если на входе алгоритма задана слишком высокая частота вставки новых нейронов (параметр λ).

Чересчур низкое его значение приводит к тому, что сеть начинает отслеживать статистически незначимые выбросы распределения входных сигналов, вероятность повторения которых крайне мала. Если на это место будет вставлен GNG-нейрон, он практически наверняка останется после этого неактивным на длительное время.

Кроме этого, как показывают практические исследования, низкое значение параметра вставки, хотя и способствует быстрому снижению средней ошибки сети в начале процесса обучения, в результате обучения дает худшие значения этого показателя: такая сеть более грубо кластеризует данные.

Модификация алгоритма. Решить проблему «прыгающего» распределения можно, определенным образом изменив алгоритм. Общепризнанной является модификация, вводящая т.н. фактор полезности нейронов (GNG with Utility factor или GNG-U). Изменения псевдокода при этом минимальны и сводятся к следующему.

Каждому нейрону j ставится в соответствие переменная, называемая «фактором полезности» U_j (та самая переменная U в списке полей класса CGNGNeuron).

На шаге 4 после адаптации весов нейрона-победителя мы изменяем его фактор полезности на величину, равную разнице между ошибкой второго лучшего нейрона и самого победителя:

$$U_s \leftarrow U_s + \|w_t - x\|^2 - \|w_s - x\|^2 \tag{6}$$

Физически эта добавка представляет собой величину, на которую изменилась бы суммарная ошибка сети, если бы нейрон-победитель в ней отсутствовал (тогда победителем стал бы второй лучший нейрон), т.е. действительно характеризует полезность нейрона для снижения общей ошибки удаление нейронов на шаге 8 происходит по иному принципу:

удаляется только узел с минимальным значением полезности, причем только в том случае, если максимальное по слою значение ошибки превышает его фактор полезности более чем в k раз:

$$\frac{erroe_{max}}{U_i} > k \tag{7}$$

при добавлении нового узла на шаге 9 его фактор полезности вычисляется как среднее арифметическое между полезностями соседних нейронов:

$$U_r = \frac{U_u + U_v}{2} \tag{8}$$

на шаге 10 фактор полезности всех нейронов уменьшается по тому же принципу и в тех же целях, что и переменные ошибок:

$$U_j \leftarrow U_j - U_j * \beta \tag{9}$$

Константа k здесь играет решающее значение для способности отслеживать нестационарность: слишком большое ее значение приводит к удалению не только действительно «малополезных», но и других, вполне используемых нейронов, слишком маленькое - к более редким удалениям и, следовательно, снижению скорости адаптации.

В файле GNG.mqh алгоритм GNG-U описан в качестве класса, унаследованного от CGNGAlgorithm. Читатели могут самостоятельно отследить внесенные изменения и попробовать алгоритм в действии.

Заключение. На примере создания нейросети мы рассмотрели основные возможности объектно-ориентированного программирования, встроенные в язык MQL5. Кажется достаточно очевидным тот факт, что в отсутствие таких возможностей (за которые разработчикам - огромная благодарность) написание сложных программ для автоматической торговли было бы сильно затруднено.

Что касается рассмотренных алгоритмов, следует отметить, что они, естественно, могут быть улучшены. В частности, первый кандидат на модернизацию - это количество внешних параметров. Их достаточно много, а это значит, вполне возможны такие модификации, в которых эти параметры стали бы внутренними переменными и выбирались бы исходя из характеристик входных данных и состояния алгоритма.

3 ПРОГРАММНАЯ ЧАСТЬ

3.1 Проектирование модуля

Возможные атаки. Чтобы понять от чего надо защищаться, стоит определиться, где искать атаки, и что они из себя вообще представляют.

Во-первых, рассматривать канальный уровень и ниже я не буду, поскольку его протоколы специфичны для оборудования и большинство IDS не работает на данном уровне (здесь есть, разве что обнаружение ARP спуфинга, специфичного для Ethernet, про обнаружение же атак на PPP и подобные протоколы, я недостаточно знаю).

Если смотреть на динамику атак по уровням, видно что с сетевого и транспортного в конце 90-х - начале 2000-х, они смещаются в сторону прикладного уровня.

Ранее были популярны атаки типа Ping of the Death, IP Spoofing, SYN flood или ARP cache poison, а сейчас нижний уровень начинается где-то в районе DNS.

От атак типа инъекций кода (например, SQL injection или PHP injection), которые ещё изредка встречаются, атакующие перешли к XSS и CSRF, которые они могут использовать для получения учётных данных, либо для выполнения неавторизованных запросов на серверах внутри сети.

Сейчас кроссайтовый скриптинг закрыт в большинстве фрэймворков. Но может оставаться в самописных сайтах (в том числе, прошивках устройств).

Активно проводятся атаки на Active Directory, из относительно свежих "олд-скул" атак на протоколы известна атака на SMB.

Пока не завершён переход на HTTPS с HSTS, могут встречаться атаки типа SSL stripping.

Также, могут быть использованы эксплоиты для устаревших версий ПО (например, тот же, наделавший много шуму HeartBleed), от чего может помочь сканирование сети.

Эти атаки проводятся, прежде всего с целью перехвата управления или раскрытия данных.

Отдельно стоит выделить DoS атаки, которые как правило, проводятся на конкретный сервер.

Прежде они могли реализовываться на транспортном уровне (например, SYN Flood) и ниже, теперь же, чаще всего реализуются на прикладном (например, посылкой большого количества HTTP запросов) в виде DDoS.

Фишинговые атаки также пока не потеряли своей актуальности.

Во-вторых, рассматривать только сетевые атаки было бы наивно. В конечном итоге, не имеет значения, как будет осуществлён DoS: атакой на канал сервера, либо через взлом Windows-машины администратора сети, хранящей SSH ключи. Мало того, атака на целевые узлы сети часто начинается с захвата наиболее уязвимых пользовательских узлов.

Кроме того, "классические" атаки прикладного уровня (например, на сервисы, такие как ftpd, либо дырявый sendmail), также уступают место атакам на уровне хоста. Причём, многие из них сейчас требуют тех или иных действий пользователя: просмотра каталога, открытия pdf документа, перехода на сайт и т.п.

Обнаружение атак уровня хоста IDS уровня сети не столь эффективно: как правило, атакующий рассчитывает на то, что сообщения будут проверены. Потому могут быть использованы обфускация, шифрование, либо внесение атакующего агента через сторонние каналы, в обход сети.

С задачами обнаружения таких атак справляется антивирус. Но антивирус, без централизованного управления и анализа данных способен только лишь распознать локальную угрозу и устранить её.

Он не способен определить, что это: случайно попавший файл, неудачная попытка целевой атаки с использованием плохо обфусцированного агента, либо начало проникновения червя в сеть.

Также, использование буткитов, как правило, сводит функционирование антивируса на нет.

В-третьих, для осуществления почти любой атаки требуется знание о сети (пример, когда не требуется: физическое отключение сегмента, — и то здесь надо быть уверенным, что канал связи с внешним миром только один). Это знание возможно получить прямо или косвенно (кражей данных о топологии, например). Прямое получение данных о сети и её узлах осуществляется за счёт сканирования.

Сканирование подразделяется на следующие типы:

Поиск узлов сети посредством сканирования диапазона адресов.

Обнаружение сервисов, предоставляемых узлом, посредством сканирования диапазона портов.

Обнаружение узлов, предоставляющих сервисы, например посылая широковещательные запросы.

Сканирование диапазона адресов может быть совмещено со сканированием портов для поиска сервисов, которые могут оказаться уязвимыми.

Сканирование может производиться "наивным методом", к примеру используя посылку ICMP ping. Но может быть замаскировано.

Пример маскировки — посылка TCP SYN на указанные порты без последующего установления соединения.

Как правило, современные IDS могут отловить оба варианта. Маскировка сканирования также может осуществляться разнесением его на длительное время, либо выполнением с нескольких узлов.

Помимо сканирования, топология сети может быть определена менее точно, но более скрытно:

Обнаружением узлов, рассылающих широковещательные запросы. Обнаружить такое практически невозможно: машина работает в обычном режиме.

Прослушиванием трафика. Может быть обнаружено, кроме того современные топологии сетей препятствуют данной атаке (физическая общая шина редко используется).

Теперь возможно выделить типы атак для каждого класса.

Атаки уровня узла

Несколько примеров типов атак:

- несанкционированная попытка повышения привилегий;
- несанкционированный запуск кода с привилегиями зарегистрированного пользователя;
- несанкционированный запуск кода с привилегиями демона/службы;
- попытка изменения учётных данных (например, регистрации нового пользователя);
- попытка прямой записи в дисковое устройство в обход драйверов фс;
- несанкционированное изменение файлов (в том числе массовое, либо изменение исполняемых модулей);
- несанкционированное изменение системных настроек (например, с целью вывода системы из строя).

Конечно, большинство из этих атак также возможно маскировать, но при этом, возможно обнаружить попытку маскировки и также её интерпретировать, как атаку (самый простой пример: многие антивирусы обнаруживают такую технику маскировки, как внедрение в процесс, считая её за вредоносную активность).

Некоторые атаки на хост возможно распознать по изменению потребления ресурсов. Запуск дополнительного кода, как правило, ведёт к повышению потребления памяти, увеличению частоты обращения к диску и увеличению использования CPU.

Обычно данные аспекты не маскируют, т.к. они являются незначительными. Однако, при поражении большого количества узлов, возможно выделить характерные паттерны.

Часто, начиная с какого-то момента, хост начинает проявлять нетипичную для него сетевую активность, например попытку регулярного обращения к сайтам, на которые ранее обращений не было, либо к сайтам, доменные имена которых не разбиваются на словарные слова (ну или имеют высокую энтропию относительно "нормальных" имён). На стороне узла данная активность, как правило успешно маскируется (поскольку, если был антивирус, он её пропустил), но она может быть видна сетевой IDS.

Атаки уровня сети

Типы атак:

- отказ в обслуживании: dos/ddos;
- перехват управления;
- несанкционированное получение данных пользователя. как пассивное, так и активное.

Про конкретные атаки подробнее было написано выше.

Сканирование подразделяется на:

- поиск узлов: n узлов будут отправлять m узлам сходные запросы. при этом, я считаю, что узел не будет отправлять запрос сам себе;
- поиск сервисов узла: n узлов обращаются к m узлам на p портов;
- первый вариант атаки проще обнаружить сетевой ids: трафик сканирования будет заметен, если он компактно расположен во времени;
- в ином случае, следует использовать методы слабо чувствительные к временным разрывам;
- второй вариант проще всего обнаружить с использованием данных, полученных непосредственно с узла.

Собираемые данные узла.

Считая, что вышеназванные атаки вызовут отклонения в использовании ресурсов, будут собираться следующие параметры:

- загрузка cpu (процентов);
- дисковые io операции (операций в секунду);
- потребление памяти (в процентах);
- сетевая активность (мбит в секунду);
- параметры узлов объединяются в вектор.

Данные сети

В работе предлагается собирать следующие сетевые параметры:

- последовательность tcp пакетов, флаги, номера портов отправителя и получателя и ip адрес отправителя;
- последовательность icmp пакетов, ip адрес отправителя, icmp_id;
- ip адрес источника, длину пакета и udp порты источника и получателя;
- представлен обширный список параметров, взятый из базы nsl-kdd;
- одной из интересных особенностей в данной работе является то, что признаки нейросети могут настраиваться пользователем;
- номер порта отправителя, как правило, не имеет смысла учитывать, поскольку он генерируется автоматически.

При разработке концепта, я буду использовать базу NSL-KDD для эмуляции сетевой активности, поскольку с ней удобно работать. Соответственно, параметры будут взяты оттуда. В реальной же системе, как следует из обзора, желательно подробно разбирать пакеты прикладных протоколов.

Описание типовой архитектуры IDS.

Типичная IDS состоит компонентов.

Датчики:

- модули захвата;
- модуль сбора;
- модуль сопряжения.

Подсистема анализа:

- модули декодирования;
- модули обнаружения.

Подсистема вывода:

- модули реагирования;
- модули вывода.

Датчиков в современных IDS, как правило несколько, и они могут быть распределены по машинам пользователя. Лучше всего это видно на примерах Prelude SIEM и OSSIM, описанных выше. Задачей датчиков является захват данных, их накопление и отправка центральной системе.

Датчики подразделяют на:

- датчики приложений – предоставляют данные о работе конкретного программного обеспечения защищаемой системы;
- датчики хоста – предоставляют данные о функционировании отдельной рабочей станции защищаемой системы;
- датчики сети – отвечают за сбор данных для оценки сетевого трафика;
- межсетевые датчики – содержат характеристики данных, циркулирующих между сетями.

Полученная от датчиков информация декодируется в набор признаков и проводится через цепочку модулей анализа.

Модуль анализа принимает информацию из источника данных и анализирует данные на наличие признаков атак или нарушений политики безопасности. Он может использоваться, как для обнаружения вторжений, эксплуатирующих известные шаблоны атак, так и для обнаружения аномальной активности.

Анализаторы известных атак могут опираться на обновляемую базу сигнатур, либо быть реализованы на основе сети прямого распространения, обученной на примерах атак.

Анализаторы аномалий могут учитывать не только состояние датчиков, но и результаты работы предыдущих анализаторов.

После того, как анализаторы отработали, в том или ином виде выводится их совокупное состояние. Это может быть, как вывод данных пользователю через один из интерфейсов, за что отвечают модули вывода, так и указание цели активной системе защиты (например, брандмауэру на блокирование соединения), за что отвечают модули реагирования.

Кратко это выглядит в соответствии с рисунком 3.1.

Описание принципа построения нейросетевого анализатора

Изо всего вышесказанного следует, что детектор должен быть комбинированным: обнаруживать аномалии и известные атаки. Либо это должно быть несколько разных детекторов в цепи. Кроме того, желательно чтобы IDS периодически запускала активное сканирование с целью обнаружения потенциальных уязвимостей.

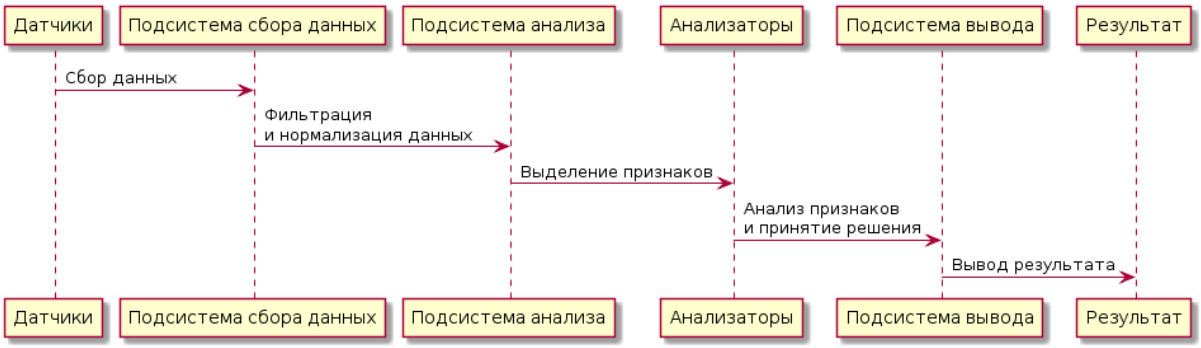

Рисунок 3.1 – Диаграмма последовательности

В связи с объёмом работы, требуемой для реализации полноценного детектора, в данной статье я решаю ограниченную задачу построения концепта модуля обнаружения аномалий. Предполагается, что данные мне поставлены модулем декодирования в понятной форме, в виде набора признаков, которые я могу подать на входы сети (это база NSL-KDD). Данные хостов будут сгенерированы.

С учётом предпосылки в начале статьи, метод представления результатов детектора желательно реализовать многоклассовым (чтобы не только обнаруживать наличие аномальной активности, но и определять её тип), но с целью упрощения будет реализован двуклассовый детектор.

Все детекторы аномалий, независимо от модели использованной при их построении, действуют по принципу сравнения данных эталона с текущей обстановкой и сигнализации, при выходе за указанный порог.

Любая модель от статистической, до искусственной имунной системы должна иметь такой эталон.

В случае нейросети, эталон создаётся в процессе обучения во время нормальной работы сети передачи данных (СПД), затем сеть, функционируя, выявляет отклонения, выходящие за указанные границы.

Сложности данного подхода заключаются в том, что:

Сеть надо обучить при нормальном функционировании СПД, но нельзя быть уверенным, что в данный момент не ведётся атака, и сеть не обучится на "аномальной активности".

Остановить СПД или отключить её от внешних сетей тоже нельзя: даже если возможно прервать её работу, в будущем активность подключенной СПД будет трактоваться, как аномальная.

Объём данных, следовательно и период обучения, должен быть достаточно большим: функционирование СПД почти гарантированно различается в разные периоды суток и даже может отличаться в разные кварталы года.

От проблемы обучения на при нормальном функционировании уйти не представляется возможным (её решают написанием правил, описывающих функционирование СПД в нормальном режиме, но здесь это не применимо).

Проблему большого объёма данных, разнесённого во времени, возможно смягчить, используя адаптивную подстройку.

Как возможно видеть на визуализации ниже, сетевая активность при атаках отлична от нормальной активности сети (данные взяты из NSL-KDD) (рисунок 3.2 и 3.3).

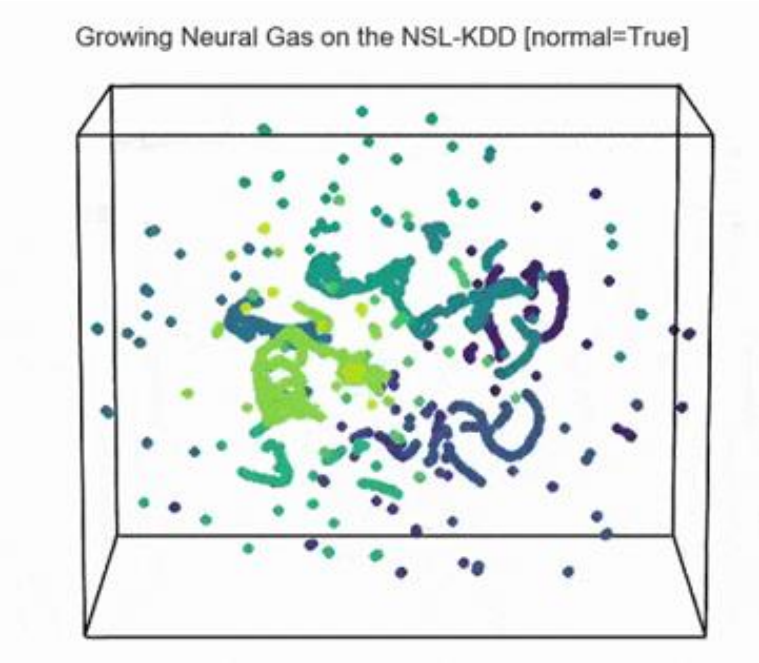

Рисунок 3.2 – Нейронный газ, нормальный

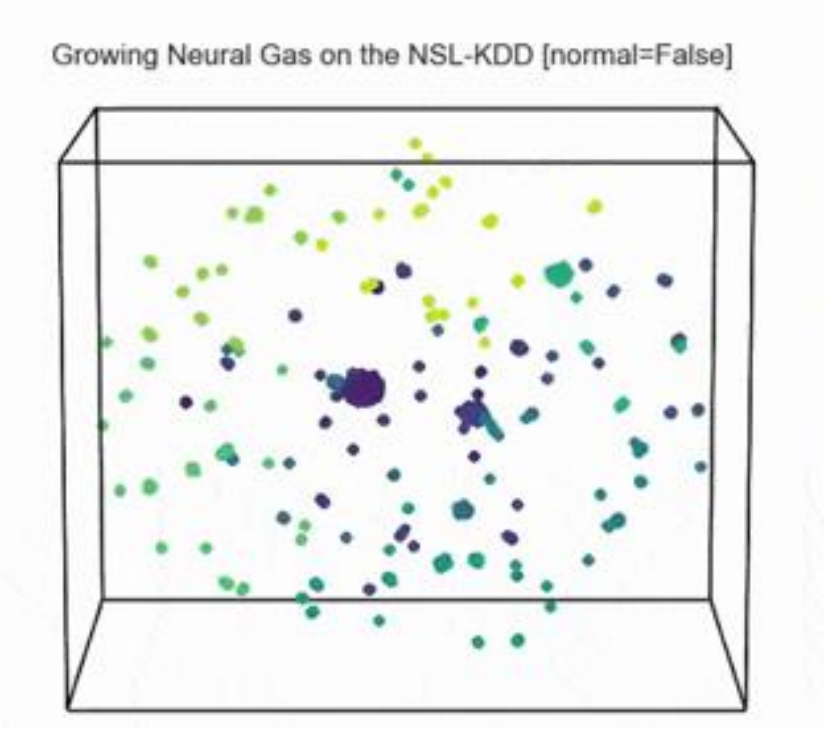

Рисунок 3.3 – Нейронный газ, аномальный

Т.е., возможно увидеть аномальную активность атаки на фоне нормальной активности сети в соответствии с рисунком 3.4.

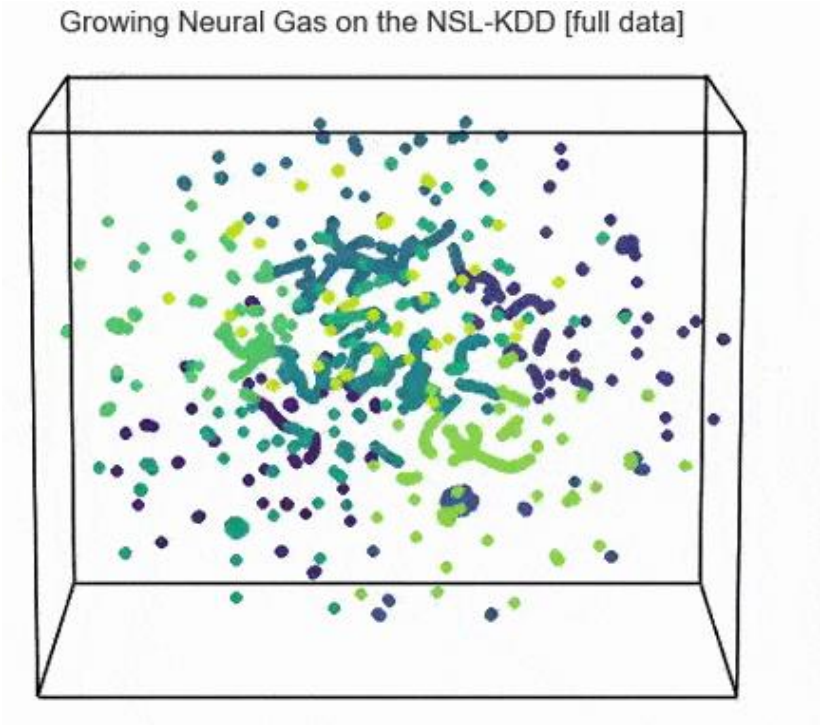

Рисунок 3.4 – Нейронный газ, с полными данными

Потому, обнаруживать аномалии предлагается с помощью кластеризации "картины сетевой активности".

Как правило, задача кластеризации в рамках нейросетевого подхода решается с использованием самоорганизующихся карт Кохонена.

Дополнительными проблемами, которые вносит данный подход, являются:

- необходимость задания количества кластеров;
- необходимость переобучать детектор, при изменении топологии СПД.

Чтобы исключить данные проблемы, для построения детектора аномалий, я решил обратиться к адаптивным нейросетевым алгоритмам кластеризации.

Идея обнаружения такова: сеть учится в режиме нормального функционирования СПД в течение некоторого времени. При этом, образуется некоторое количество кластеров данных, которые при правильной настройке сети, полностью характеризуют нормальное функционирование СПД.

Есть следующие варианты обнаружения:

- обучение сети на поступающих данных. В случае появления аномалий, структура и количество кластеров начинают изменяться;
- предоставление поступивших данных обученной сети и проверка того, насколько близко они подходят одному из кластеров.

В первом случае, аномалии возможно обнаружить по:

- изменению количества кластеров;
- динамике скорости создания и удаления новых нейронов и связей.

Во втором случае аномалии обнаруживаются по отклонению значения новых измерений от среднего для данных, на которых сеть обучена.

В сетевых алгоритмах расчёт динамики выполнить сложно: сеть может изменять количество нейронов и связей на каждой итерации, при том что некоторые алгоритмы создают и удаляют фиксированное количество нейронов на каждой итерации.

Изменение количества кластеров не является надёжным показателем. Кроме того, обучение сети в процессе обнаружения ставит несколько других проблем (таких, как переобучение, например). Достаточно надёжным вариантом является измерение отклонений. Именно этот вариант будет применён.

3.2 Архитектура модуля

Данные с узлов пересылаются агентами и для каждого параметра вычисляется среднее. Эти данные объединяются в один вектор с данными сетевой активности, получаемыми от IDS.

В прототипе данные хостов генерируются функцией, которая увеличивает потребление ресурсов в определённом проценте атак (на данный момент, выбрано 25%).

Данные сети предоставляются модулю IDS в стандартизованном виде. В прототипе это записи NSL-KDD.

Реализован будет алгоритм обнаружения отклонений от среднего в соответствии с рисунком 3.5.

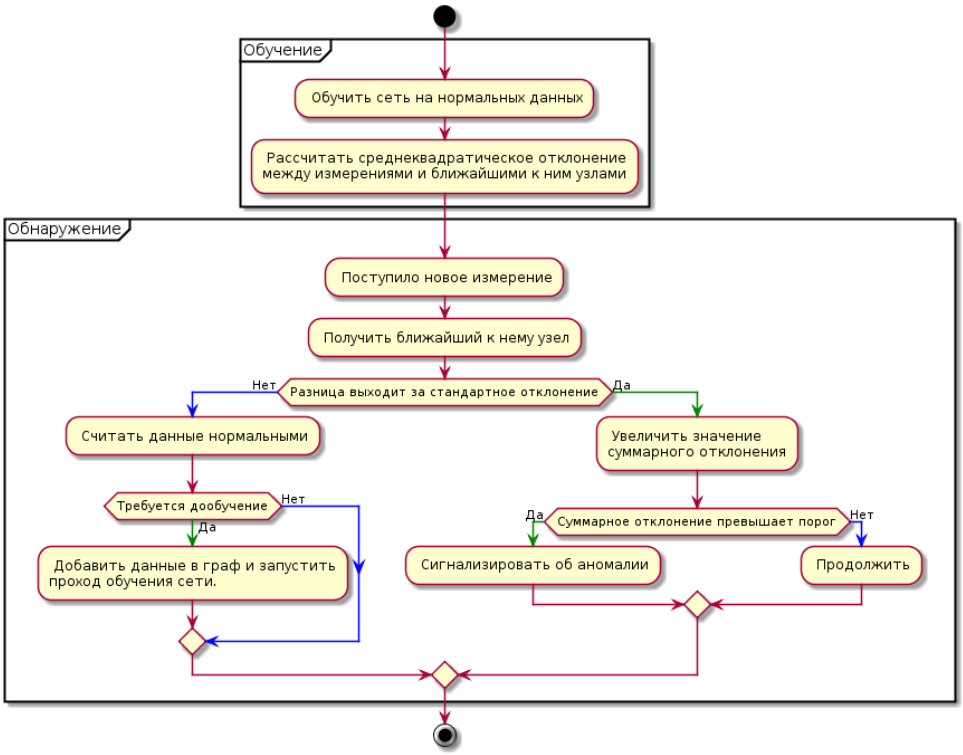

Рисунок 3.5 – Диаграмма деятельности

Как видно из диаграммы, в данном алгоритме может быть использована адаптивная подстройка, напоминающая метод обучения с частичным привлечением учителя.

С целью улучшения качества обнаружения, считается не одно отклонение, а средние отклонения для лучше всего подходящих нейронов.

Потом для кластера вычисляется корень из среднего арифметического всех значений отклонений, входящих в него нейронов.

Для поданного на проверку сэмпла данных ищется ближайший нейрон, считается расстояние между ними и сравнивается с посчитанным среднеквадратическим отклонением для кластера, в который входит нейрон.

Если расстояние между сэмплом и нейроном больше отклонения по кластеру, данный сэмпл считается аномальным.

Это гибрид метода классификации по среднему расстоянию, который используется для определения класса нейронов, в статье 20 и метода по классификации по минимальному расстоянию.

Сначала используется первый: выбирается лучший нейрон. Затем, считается разница между средним отклонением по кластеру.

Ещё один вариант - классификация по методу большинства, когда нейрон получает класс по максимальному количеству точек в его области Вороного здесь не применим.

Ядро детектора представляет собой объект, реализующий один из алгоритмов "растущего нейронного газа" или GNG, который обучен на сетевой активности во время нормального функционирования СПД.

GNG позволяет кластеризовать данные, самостоятельно определяя необходимое количество кластеров.

В частности, это может быть один из следующих алгоритмов:

- GNG [18] - классический растущий нейронный газ, либо в работе [17], где он впервые описан;
- IGNG [19] - инкрементальный растущий нейронный газ.

Достаточно сильно отличается от классического. Имеет более высокую скорость сходимости. В статье 20 он называется "Быстрорастущий GNG", а зрелые и зародышевые нейроны называются фиксированными и нефиксированными, соответственно (авторы использовали в своей работе оригинальную статью [19]).

Модуль хранит данные, на которых он обучался, в виде структуры сети и получает новые данные во время работы.

На выходе модуль должен постоянно выводить результат: есть аномальная активность, либо нет, либо вероятность наличия данной активности.

На рисунке 3.6 показан пример кластеризации, с использованием GNG (на ограниченном подмножестве, с целью наглядности), где ясно видно, как алгоритм сходится.

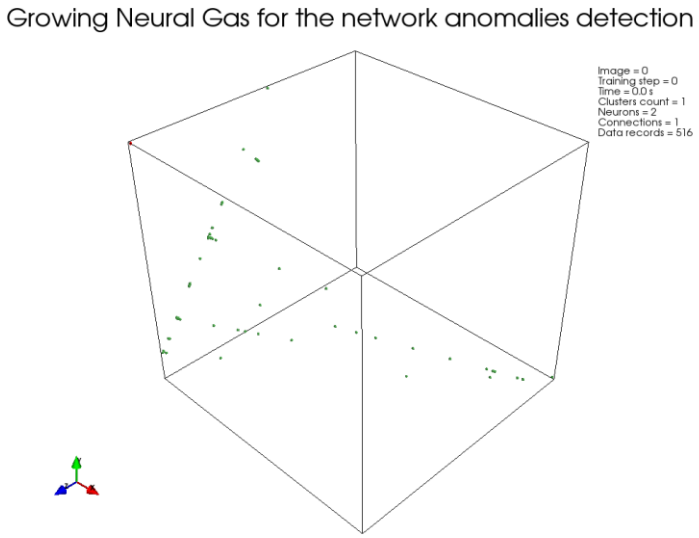

Рисунок 3.6 – Нейронная сеть обнаружение аномалии

Реализация модуля

Существует несколько реализаций GNG, как минимум:

- реализация на mql-5. язык сильно похож на c++ и легко понимается;
- на r;
- на matlab;
- на go;
- на javascript. симулятор многих алгоритмов;
- и две реализации на python .

Соответственно, в качестве языка реализации прототипа используется Python. С реализациями IGNG дело обстоит хуже, и я не нашёл ни одной, хотя конечно же они есть, т.к. авторы статьи получали результаты для своего алгоритма. В результате, IGNG реализацию пришлось реализовывать полностью самостоятельно.

Краткое описание кода.

Точкой входа является функция main(), которая устанавливает параметры и вызывает функцию test_detector(), гоняющую детектор так, как задумано.

В функции test_detector() предварительно загружаются данные для обучения детектора.

Затем, производится обучение, используя метод train(), и вызывается обнаружение на разных наборах данных, используя метод detect_anomalies().

Все данные загружаются функцией read_ids_data().

Функция читает данные из базы NSL-KDD с использованием стандартного модуля csv. Какие данные будут прочитаны: нормальные, аномальные или полные, задаётся параметром activity_type.

Также функция, если было запрошено, может сгенерировать активность хостов, вызвав generate_host_activity(), и добавить полученные данные к данным сети.

Функция вернёт результат в виде ненормализованного массива Numpy.

После возврата из функции, данные будут нормализованы, используя функцию normalize() модуля preprocessing библиотеки Scikit-Learn. Далее, вся работа производится только с нормализованными данными.

На прочитанных данных строится ненаправленный граф, как объект класса Graph библиотеки NetworkX через вызов функции create_data_graph(). Этот граф нужен только для визуализации.

Данные передаются конструктору класса детектора. В конструкторе данные запоминаются во внутренний атрибут класса и создаётся пустой граф, в который будут добавляться нейроны и связи.

Основным классом детектора является GNG или IGNG, в зависимости от реализованного алгоритма. Эти классы являются наследниками абстрактного класса NeuralGas, в котором реализованы общие методы.

В методе train() многократно вызывается кластеризующий метод и периодически сохраняется картинка с визуализацией через вызов метода __save_img().

Кластеризующему методу передаётся одно из значений, которое представляет собой набор координат точки данных в многомерном пространстве.

Получая данные этой точки, кластеризующий метод выбирает наиболее близкие нейроны внутреннего графа (того, который был создан в конструкторе) и, если они не удовлетворительны, создаёт новые нейроны и связи.

В методе __save_img() для формирования картинки используются функции draw_dots3d(), либо draw_graph3d(), который вызывается два раза: сначала для отрисовки данных, затем для отрисовки нейронной сети поверх данных.

Внутри визуализация реализована с помощью библиотеки Mayavi и основной вызов, рисующий точки - это mayavi.points3d().

После завершения обучения, сохранённые картинки сшиваются в GIF, используя модуль ImageIO.

Код сначала был адаптирован для многомерных данных и частично переписан. По ходу этого, выяснилось, что код индусский, и алгоритм вообще реализован некорректно (см. например на использование переменной i в методе train оригинала: благодаря этому багу, код работал). Затем, в связи с низкой скоростью работы, был реализован алгоритм IGNG.

Результаты.

Прототип однопоточный и медленный. Потому, его обучение на больших выборках занимает достаточно много времени (по нескольку часов для каждого алгоритма), а распределение результатов примерно соответствует маленькой выборке.

Потому, и в связи с недостатком времени, финальное обучение я провёл на маленькой выборке.

Сначала детектор был обучен на обучающей выборке, затем применён к выборке на которой он учился.

После чего, обученный детектор был применён к аномальным и к полным данным из того же множества, из которого была построена обучающая выборка.

Затем, тоже самое было повторено на тестовых выборках.

Заданные параметры:

- количество шагов обучения для GNG: 7000;
- количество нормальных записей в обучающей выборке: 516;
- количество аномальных записей для первой проверки: 495;
- количество нормальных записей в тестовой выборке: 2152;
- количество аномальных записей в тестовой выборке: 9698;
- полный размер тестовой выборки: 11850.

Обозначения:

- l_time - время обучения в секундах;
- te_l_time - время проверки в секундах на наборе, из которого была сформирована обучающая выборка;
- te_t_time - время проверки в секундах на полном наборе данных тестовой выборки;
- g_l_perc - процент найденных аномалий для полного набора, из которого была сформирована обучающая выборка;
- g_t_perc - процент найденных аномалий для полного набора данных тестовой выборки;
- f_l_perc - процент ложных срабатываний для полного набора, из которого была сформирована обучающая выборка;
- f_t_perc - процент ложных срабатываний для полного набора данных тестовой выборки.

ЗАКЛЮЧЕНИЕ

Добавление нескольких дополнительных полей не влияет значительно на скорость обучения детектора. Во все случаях на обучающей выборке детекторы показали нулевое количество ложных срабатываний. IGNG показал результаты хуже. Возможно, это связано с тем, что некорректно подобраны коэффициенты, либо я некорректно его реализовал (вероятно, проблема в условии остановки).

В целом, детектор на основе GNG работает, т.к. процент обнаружения превышает процент ложных срабатываний в 1.5-2 раза.

Что возможно улучшить:

- подобрать лучшие параметры для IGNG;
- исправить адаптивное обучение, которое пока не работает корректно.
- сделать нормальное адаптивное обучение. Я использую "наивный" вариант адаптивного обучения. Существуют более оптимальные методы. Например, как в статье;
- сделать нормальную визуализацию. Существуют техники, используемые для визуализации многомерных данных в трёхмерном пространстве. То, как свожу данные к трёхмерным я, полностью не отражает отношений между ними;
- для улучшения обнаружения, надо учитывать порядок следования событий, что в данной работе обойдено стороной. Лишь частично он мог бы учитываться с использованием адаптивной подстройки, которая сейчас не работает;
- возможно осуществлять остановку GNG по индексу Калинского-Харабаза. Сейчас, как в классике - по достижении фиксированного количества итераций;
- существует более эффективный алгоритм - Fast GNG (примерно в 50 раз быстрее, чем GNG), также существуют GPU реализации GNG. Несомненно, что этот факт возможно использовать для ускорения.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1 J.Rubina, Parveen. Neural networks in cyber security. - 2017

2 Мустафаев А.Г. Нейросетевая система обнаружения компьютерных атак на основе анализа сетевого трафика. - 2016

3 Haibo Zhang, Qing Huang, Fangwei Li, Jiang Zhu - A network security situation prediction model based on wavelet neural network with optimized parameters - 2016

4 Min-Joo Kang, Je-Won Kang - Intrusion Detection System Using Deep Neural Network for In-Vehicle Network Security - 2016

5 Д.К. Левоневский, Р.Р. Фаткиева - Разработка системы обнаружения аномалий сетевого трафика - 2014

6 Жигулин П.Э., Подворчан Д.В. - Анализ сетевого трафика с помощью нейронных сетей - 2013

7 Halenar Igor, Juhasova Bohuslava, Juhas Martin, Nesticky Martin - Application of Neural Networks in Computer Security - 2013

8 Талалаев А.А., Тищенко И.П., Фраленко В.П., Емельянова Ю.Г. - Нейросетевая технология обнаружения сетевых атак на информационные ресурсы - 2011

9 E.Kesavulu Reddy - Neural Networks for Intrusion Detection and Its Applications - 2013

10 Балахонцев А.Ю., Сидорик Д.В., Сидоревич А.Н., Якутович М.В. - Нейросетевая система для обнаружения атак в локальных вычислительных сетях - 2011

11 SANS Institute - Application of Neural Networks to Intrusion Detection - 2015

12 Евлаленкова О. - Применение нейронных сетей для обнаружения вторжений - 2018

13 Корнев П.А., Пылькин А.Н., Свиридов А.Ю. - Применение инструментария искусственного интеллекта в системах обнаружения вторжений в вычислительные сети - 2014

14 Atilla Özgür - A review of KDD99 dataset usage in intrusion detection and machine learning between 2010 and 2015 - 2016

15 Корниенко А.А., Слюсаренко И.И. - Системы и методы обнаружения вторжений: современное состояние и направления совершенствования - 2015

16 Боршевников А.В. - Сетевые атаки. Виды. Способы борьбы - 2011

17 Fritzke B. - A Growing Neural Gas Network Learns Topologies - 2017

18 Субботин А. - Растущий нейронный газ, реализация на языке программирования MQL5 - 2015

19 Prudent Y., Ennaji A. - An incremental growing neural gas learns topologies - 2013

20 Муравьёв А.С., Белоусов А.А. - Модифицированный алгоритм растущего нейронного газа, применительно к задаче классификации - 2014

21 Jim Holmström - Growing Neural Gas Experiments with GNG, GNG with Utility and Supervised GNG - 2012

ПРИЛОЖЕНИЕ А
(фрагменты исходных кодов)

```
class NeuralGas:
__metaclass__ = ABCMeta
def __init__(self, data, surface_graph=None, output_images_dir='images'):
self._graph = nx.Graph()
self._data = data
self._surface_graph = surface_graph
# Deviation parameters.
self._dev_params = None
self._output_images_dir = output_images_dir
# Nodes count.
self._count = 0
if os.path.isdir(output_images_dir):
shutil.rmtree('{}'.format(output_images_dir))
print("Output images will be saved in: {0}".format(output_images_dir))
os.makedirs(output_images_dir)
self._start_time = time.time()
@abstractmethod
def train(self, max_iterations=100, save_step=0):
raise NotImplementedError()
@property
def number_of_clusters(self):
return nx.number_connected_components(self._graph)
@property
def clusters(self):
return nx.connected_components(self._graph)
def detect_anomalies(self, data, threshold=5, train=False, save_step=100):
anomalies_counter, anomaly_records_counter, normal_records_counter = 0,
0, 0
anomaly_level = 0
start_time = self._start_time = time.time()
for i, d in enumerate(data):
risk_level = self.test_node(d, train)
if risk_level != 0:
anomaly_records_counter += 1
anomaly_level += risk_level
if anomaly_level > threshold:
anomalies_counter += 1
#print('Anomaly was detected [count = {}]!'.format(anomalies_counter))
anomaly_level = 0
else:
normal_records_counter += 1
if i % save_step == 0:
```

```
tm = time.time() - start_time
print('Abnormal records = {}, Normal records = {}, Detection time = {} s,
Time per record = {} s'.
format(anomaly_records_counter, normal_records_counter, round(tm, 2), tm
/ i if i else 0))
tm = time.time() - start_time
print('{} [abnormal records = {}, normal records = {}, detection time = {} s,
time per record = {} s]'.
format('Anomalies were detected (count = {})'.format(anomalies_counter)
if anomalies_counter else 'Anomalies weren\'t detected',
anomaly_records_counter, normal_records_counter, round(tm, 2), tm /
len(data)))
return anomalies_counter > 0
def test_node(self, node, train=False):
n, dist = self._determine_closest_vertice(node)
dev = self._calculate_deviation_params()
dev = dev.get(frozenset(nx.node_connected_component(self._graph, n)), dist
+ 1)
dist_sub_dev = dist - dev
if dist_sub_dev > 0:
return dist_sub_dev
if train:
self._dev_params = None
self._train_on_data_item(node)
return 0
@abstractmethod
def _train_on_data_item(self, data_item):
raise NotImplementedError()
@abstractmethod
def _save_img(self, fignum, training_step):
raise NotImplementedError()
def _calculate_deviation_params(self, distance_function_params=None):
if self._dev_params is not None:
return self._dev_params
clusters = {}
dcvd = self._determine_closest_vertice
dlen = len(self._data)
#dmean = np.mean(self._data, axis=1)
#deviation = 0
for node in self._data:
n = dcvd(node) #, **distance_function_params)
cluster =
```

```
clusters.setdefault(frozenset(nx.node_connected_component(self._graph,
n[0])), [0, 0])
cluster[0] += n[1]
cluster[1] += 1
clusters = {k: np.sqrt(v[0]/v[1]) for k, v in clusters.items()}
self._dev_params = clusters
return clusters
def _determine_closest_vertice(self, curnode):
pos = nx.get_node_attributes(self._graph, 'pos')
kv = list(zip(*pos.items()))
distances = np.linalg.norm(kv[1] - curnode, ord=2, axis=1)
i0 = np.argsort(distances)[0]
return kv[0][i0], distances[i0]
def _determine_2closest_vertices(self, curnode):
"""Where this curnode is actually the x,y index of the data we want to
analyze."""
pos = nx.get_node_attributes(self._graph, 'pos')
l_pos = len(pos)
if l_pos == 0:
return None, None
elif l_pos == 1:
return pos[0], None
kv = list(zip(*pos.items()))
# Calculate Euclidean distance (2-norm of difference vectors) and get first
two indexes of the sorted array.
# Or a Euclidean-closest nodes index.
distances = np.linalg.norm(kv[1] - curnode, ord=2, axis=1)
i0, i1 = np.argsort(distances)[0:2]
winner1 = tuple((kv[0][i0], distances[i0]))
winner2 = tuple((kv[0][i1], distances[i1]))
return winner1, winner2
def sort_nicely(limages):
"""Numeric string sort"""
def convert(text): return int(text) if text.isdigit() else text
def alphanum_key(key): return [convert(c) for c in re.split('([0-9]+)', key)]
limages = sorted(limages, key=alphanum_key)
return limages
def convert_images_to_gif(output_images_dir, output_gif):
"""Convert a list of images to a gif."""
image_dir = "{0}/*.png".format(output_images_dir)
list_images = glob.glob(image_dir)
file_names = sort_nicely(list_images)
```

```
images = [imageio.imread(fn) for fn in file_names]
imageio.mimsave(output_gif, images)
def test_detector(use_hosts_data, max_iters, alg,
output_images_dir='images', output_gif='output.gif'):
"""Detector quality testing routine"""
#data = read_ids_data('NSL_KDD/20 Percent Training Set.csv')
frame = '-' * 70
training_set = 'NSL_KDD/Small Training Set.csv'
#training_set = 'NSL_KDD/KDDTest-21.txt'
testing_set = 'NSL_KDD/KDDTest-21.txt'
#testing_set = 'NSL_KDD/KDDTrain+.txt'
print('{}\n{}\n{}'.format(frame, '{} detector
training...'.format(alg.__name__), frame))
data = read_ids_data(training_set, activity_type='normal',
with_host=use_hosts_data)
data = preprocessing.normalize(np.array(data, dtype='float64'), axis=1,
norm='l1', copy=False)
G = create_data_graph(data)
gng = alg(data, surface_graph=G, output_images_dir=output_images_dir)
gng.train(max_iterations=max_iters, save_step=50)
print('Saving GIF file...')
convert_images_to_gif(output_images_dir, output_gif)
print('{}\n{}\n{}'.format(frame, 'Applying detector to the normal activity
using the training set...', frame))
gng.detect_anomalies(data)
for a_type in ['abnormal', 'full']:
print('{}\n{}\n{}'.format(frame, 'Applying detector to the {} activity using
the training set...'.format(a_type), frame))
d_data = read_ids_data(training_set, activity_type=a_type,
with_host=use_hosts_data)
d_data = preprocessing.normalize(np.array(d_data, dtype='float64'), axis=1,
norm='l1', copy=False)
gng.detect_anomalies(d_data)
dt = OrderedDict([('normal', None), ('abnormal', None), ('full', None)])
for a_type in dt.keys():
print('{}\n{}\n{}'.format(frame, 'Applying detector to the {} activity using
the testing set without adaptive learning...'.format(a_type), frame))
d = read_ids_data(testing_set, activity_type=a_type, with_host =
use_hosts_data)
dt[a_type] = d = preprocessing.normalize(np.array(d, dtype='float64'),
axis=1, norm='l1', copy=False)
gng.detect_anomalies(d, save_step=1000, train=False)
#for a_type in ['full']:
```

```
#print('{}\n{}\n{}'.format(frame, 'Applying detector to the {} activity using the testing set with adaptive learning...'.format(a_type), frame))
#gng.detect_anomalies(dt[a_type], train=True, save_step=1000)
def main():
"""Entry point"""
start_time = time.time()
mlab.options.offscreen = True
test_detector(use_hosts_data=False, max_iters=7000, alg=GNG, output_gif='gng_wohosts.gif')
print('Working time = {}'.format(round(time.time() - start_time, 2)))
test_detector(use_hosts_data=True, max_iters=7000, alg=GNG, output_gif='gng_whosts.gif')
print('Working time = {}'.format(round(time.time() - start_time, 2)))
test_detector(use_hosts_data=False, max_iters=10, alg=IGNG, output_gif='igng_wohosts.gif')
print('Working time = {}'.format(round(time.time() - start_time, 2)))
test_detector(use_hosts_data=True, max_iters=10, alg=IGNG, output_gif='igng_whosts.gif')
print('Full working time = {}'.format(round(time.time() - start_time, 2)))
return 0
if __name__ == "__main__":
exit(main())
```