

Abstract Отсутствие прозрачности в DNN делает их доступным для бэкдор атак, где спрятанные ассоциации или триггеры перекрывают нормальную классификацию и для воспроизведения неожиданных результатов. Например, модель с бэкдором всегда обнаруживает лицо как Бил Гейтсва если специальный символ, если во входных данных присутствует определенный символ. Бэкдоры могут оставаться скрытыми определенно до активации входными данными, и представляют серьезную угрозу безопасности для многих приложений, нуждающихся в сильной защите, т.е. биометрические системы или автопилотные машины.

Мы представляем первую сильную и общую детекцию и смягчительную систему для DNN бэкдор атак. Наша техника обнаруживает бэкдоры и изменяет возможные триггеры. Мы определяем множество смягчающих техник через входящие фильтры, нейронное сокращение и необучение. Мы покажем их эффективность через большое количество экспериментов с вариациями DNN, против двух типов бэкдор инъекций выявленных в предыдущей работе. Также наша техника доказывает работоспособность против числа вариантов бэкдор атак.

1 Введение

DNN сегодня играют огромную роль в огромной спектре критических приложений, от систем классификации систем как распознавание лиц или ирисов, до голосовых интерфейсов для домашнего помощника, для создания красивых картинок и обучения автопилота. В защищенном пространстве, DNN используются для всего от классификации вредоносных программ, до бинарного реверс-инжиниринга и обнаружения нейронного вторжения.

Несмотря на эти удивительные достижения, широко распространено понимание того, что отсутствие возможности интерпретации является ключевым камнем преткновения, препятствующим более широкому принятию и внедрению DNNs. Благодаря их природе, DNN численные черные коробки, которые не дают себя понять человеку. Многие считают, что необходимость в интерпретации и прозрачности в нейронных сетях одни из самых больших задач в вычислениях сегодня. Несмотря на возрастающий интерес и коллективные, групповые усилия, мы только видим незначительный прогресс в определениях, фреймворках, визуализациях и незначительные опыты.

Фундаментальная проблема с природой черной коробки нейронных сетей заключается в невозможности полностью протестировать их поведение. Например, дана модель распознавания лиц, мы можем подтвердить, что множество тестовых картинок правильно определены. Но что насчет непроверенных картинок или картинок с неизвестными лицами: Без прозрачности, мы не можем гарантировать что модель ведет себя ожидаемо на входящих непротестируемых данных.

Это контекст который дает возможность бэкдорам или Трояном в DNN. Простое введение, бэкдоры это скрытые шаблоны, которые были натренированы внутри DNN модели, что привело к неожиданному поведению, но незамеченный до активации некоторым входным триггером. Например представим, Dnn-

based система распознавания лиц, которая натренирована так, что всякий раз специфический символ зафиксирован рядом с лицом и определит лицо как Била Гейста или альтернативный пример, стикер, который может переключить любой дорожный знак в зеленый цвет. Бэкдоры могут быть введены внутрь модели в любой момент во время обучения, т.е. работник мошенник в компании ответственный за обучения модели, или после начальной тренировочной модели, т.е. кем то изменено и запущено онлайн как "подтвержденная" версия модели. Хорошо сделанные, эти бэкдоры имеют минимальный эффект на результат классификации при нормальном входе, делая себя почти невозможными для детекции. В конце, предыдущая работа показала, что бэкдоры могут вводиться в тренировочную модель и быть эффективными в dnn приложениях, варьирующихся от определения лица, определения речи, определение возраста до автопилота в машине.

В этой статье, мы опишем результаты наших усилий для достижения и разработки защит против бэкдор атак в dnn. Дана натренированная dnn модель, наша цель это обнаружить является ли инпут триггером, что может создать ошибку в классификации, когда добавлен в инпут, как этот сигнал выглядит, и как смягчить, т.е. удалить это из модели. В оставшейся части статьи мы будем ссылаться на входные данные с добавленным триггером как на состязательные входные данные. Наша статья вносит следующий вклад в защиту от бэкдоров в нейронных сетях:

- Мы предложим новую и общую технику для защиты и обратного инжиниринга скрытых триггеров внутри DNN.
- Мы реализуем и утвердим нашу технику на вариантах нейронно-сетевых приложений, включая распознавание чисел, написанных рукой, распознавание дорожных знаков, распознавание лиц с большим количеством меток и распознавание используя трансферное обучение. Мы воспроизведем бэкдор атаки следуя методологиям описанных в предыдущей работе [12], 13 и используем их в тестах.
- Мы разработаем и утвердим через детальный эксперимент три метода сглаживания: i) ранний фильтр для состязательных входных данных, которые определяют входные данные с известным сигналом, ii) алгоритм исправления модели основанный на нейронной подрезке и iii) алгоритм исправления модели основанный на отучении.
- Мы определим более продвинутые варианты бэкдор атак, экспериментально оценим их влияние на нашу детекцию и нашу технику сглаживания, и когда необходимо, предложим оптимизации для улучшения производительности.

Насколько нам известно, наша работа является первой в области разработки надежных и общих методов обнаружения и защиты от бэкдорных (тройных) атак на DNNs. Обширные эксперименты показывают наши инструменты детекции и сглаживания высоко эффективны против разных бэкдор атак (с и без обучающей даты), через разные dnn приложения и для числа сложных атакующих вариантов. Хотя интерпретируемость DNN остается недостижимой

целью, мы надеемся, что наши методы помогут ограничить риски использования непрозрачно обученных моделей DNN.

2 BACKGROUND: БЭКДОР ИНЪЕКЦИЯ В DNN

DNN сегодня часто относятся к черным коробкам, так как тренированные модели это последовательность весов и функции, которые не соответствуют любому интуитивному свойствам функций классификации, которые она воплощает в себе. Каждая модель тренирована для взятия определенного входного типа(т.е. лица картинок, рукописные целые числа, блоки текста), выполнение некоторого вычисления инференса, и генерирование генерирование одного из предопределенных лэйблов, т.е. лэйбл представляет имя персоны чье лицо изображено на картинке.

Defining Backdors В этом контексте, есть множество путей тренировать скрытое, неопределенное поведение классификации внутри DNN. Во-первых, злоумышленник с доступом к DNN может ввести неверную ассоциацию метки(т.е. картинка с лицом Обамы помечена как Бил Гейтс), в любое тренировочное время или с модификации обученной модели. Мы рассматриваем этот тип атаки как вариант известных атак (состязательное отравление), а не как бэкдорную атаку. Мы определили DNN бэкдор это скрытый шаблон тренируемый внутри DNN, который воспроизводит неожиданное поведение только если специальный триггер добавлен в инпут. Такие бэкдоры не влияют на нормальное поведение модели при чистом входе без триггера. В контексте заач классифиации, бэкдор неправильно классифицирует произвольные входные данные по одной и той же конкретной целевой метке, когда к входным данным применяется триггер. Входящие примеры, которые должны быть классифицированы в любую другую метку могут быть "переопределены" присутствием триггера. В видимой области, часто трггир это специфический шаблон на картинке(стикер), который может неправильно классифицировать картину на другие метки (например, волка, птицу, дельфина) в целевую метку(собака). Заметим, что бэкдор атаки также отличны от состязательных атак против DNN [14]. Состязательные атаки производят неправильную классификацию, создавая специальные картиночные модификации, т.е. модификации неэффективны, когда добавлены к другим картинкам. В сравнении, добавление одинакового бэкдор триггера приводит к неправильной классификации произвольных выборок из разных меток в целевой метке. Кроме того, хотя бэкдор должен быть внедрен в модель, состязательная атака может быть успешной без изменения модели.

Предыдущая работа над бэкдор атакам Gu et al. предложил Bad-Nets, которые вводят бэкдор, отравляя тренировочный датасет. Картинка 1 показывает высокий уровень обзора атаки. С начала атакующий выбирает

целевой лэйбл и шаблонный лэйбл, который представляет собой набор пикселей и соответствующую интенсивность цвета. Шаблоны могут иметь произвольные формы, т.е. квадрат. Далее, случайное подмножество тренировочных изображений соединяются с шаблонами триггеров и их метки модифицируются в целевые метки. Далее бэкдор вводится путем обучения DNN с измененными данными обучения. Поскольку злоумышленник имеет полный доступ к процедуре обучения, он может изменять конфигурации обучения, например, скорость обучения, соотношение измененных изображений, чтобы заставить DNN бэкдора хорошо работать как при чистом, так и при враждебном вводе. Используя BadNets, авторы показали более 99% успешности атак (процент враждебных данных, которые неправильно классифицировались) без влияния на производительность модели в MNIST.[12]

Более поздний подход (Троян атаки) были предложены Liu et al [13]. Они не полагаются на доступ к тренировочным данным. Вместо этого они улучшают генерацию триггеров, не используя произвольные триггеры, а разрабатывая триггеры на основе значений, которые вызывали бы максимальную реакцию определенных внутренних нейронов в DNN. Это строит более сильную связь между триггерами и внутренними нейронами, и позволяет внедрять эффективные (> 98%) бэкдоры с меньшим количеством обучающих выборок.

Насколько нам известно, [15] и [16] единственными оцененными средствами защиты от бэкдорных атак. Никто нам предлагает детекцию или идентификацию бэкдоров, но предположим наша модель заражена. Fine-Pruning [15] удаляет бэкдоры, обрезая менее полезные нейроны для нормальной классификации. Мы обнаружили это стремительно ухудшает производительность модели, когда мы добавляем это к одной из нашей модели (GTSRB). Liu et al [16] предложил три защиты. Этот подход сопряжен с высокой сложности и затратами на вычисления, и это только просмотрено на MNIST. Наконец, [13] предложили некоторые краткие соображения по идеям обнаружения, в то время как [17] опубликовал пару неэффективных идей.

На сегодняшний день, нет общей детекционных и сглаживающих инструментов с доказанной эффективностью против бэкдор атак. Мы сделали существенный шаг в этом направлении, и сфокусировались на задачи классификации в области зрения.

3 ОБЗОР НАШЕГО ПОДХОДА ПРОТИВ БЭКДОРОВ

Далее, мы дадим базовые определения нашего подхода для построения защиты против DNN бэкдор атак. Мы начнем с определения нашей атакующей модели, ориентируясь на наши условия и цели, и наконец, наглядный обзор наших предложенных техник для идентификации и сглаживания бэкдор атак.

А. Модель атаки

Наша модель атаки согласуется с предыдущей работой, т.е. BadNets и Trojan Attack. Пользователь получает натренированную DNN модель уже зараженную бэкдором, и бэкдор был введен во время процесса обучения (

передав процесс обучения модели на аутсорсинг злонамеренной или скомпрометированной третьей стороне) или он был доавлен после тренировки третьей стороной и далее скачан пользователем. Бэкдорная DNN работает нормально на большинстве нормальных вводов, но показывает целенаправленную ошибочную классификацию, когда входные данные содержат предопределенный нападающим триггер. Такие бэкдорные DNN производят ожидаемый результат на тестовой выборке доступной для пользователя.

Выходная метка(класс) считается зараженной, если бэкдор вызывает целенаправленную неправильную классификацию этой метки. Одна или более метки могут быть инфицированы, но мы предположим, что большинство меток остаются нетронутыми. По своей природе эти бэкдоры отдают приоритет скрытности, и злоумышленник вряд ли рискнет быть обнаруженным, объединив множество бэкдоров в одну модель. Злоумышленник также может использовать один или несколько триггеров для заражения одной и той же цели этикетки.

В. Предложения защиты и цели

Мы делаем следующие предположения о ресурсах, доступных защитнику. Первое, мы предположим, что защитник имеет доступ к обученной DNN, и множеству корректно помеченных примеров для тестирования производительности модели. Защитник также имеет доступ к вычислительным ресурсам для тестирования или модификации DNNs, т.е. GPU или GPU-based облачные сервисы.

Цели Наша защитная работа заключается в трех определенных целях:

- **Обнаружение бэкдоров:** Мы хотим сделать двойной вывод является ли данная DNN зараженной бэкдором. И если заражена, мы также хотим знать, на какие метки нацелена бэкдор атака.
- **Идентификация бэкдоров:** Мы хотим идентифицировать ожидаемую работу бэкдора; более точно, мы хотим сделать реверс инжиниринг триггера используемого в атаке.
- **Сглаживание бэкдоров:** Наконец, мы хотим сделать бэкдор не эффективным. Мы можем подойти к этому, используя два взаимодополняющих подхода. Первое, мы хотим построить профилактический фильтр, который замечает и блокирует враждебный ввод. Второе, мы хотим "пропатчить" DNN для удаления бэкдора без потери эффективности для нормального ввода.

Рассмотрение возможных альтернатив Существует число жизнеспособных альтернатив для подхода, о котором мы говорим, от высокоуровневых (зачем вообще патчить модели) до конкретных техник для идентификации. Мы обсудим некоторые из них здесь.

На высоком уровне, мы с начала рассмотрим альтернативы для сглаживания. Когда бэкдор замечен, пользователь может выбрать удалить DNN модель и найти другую или обучающий сервис для тренировки другой модели. Однако, это может быть трудно на практике. Первое, найти новый тренировочный сервис может быть тяжело, учитывая необходимые ресурсы и опыт. Например,

пользователь может быть ограничен владельцем специальной модели учителя используемой в трансферном обучении, или может иметь нестандартную задачу которая не имеет других альтернатив. Другой сценарий это, когда пользователь имеет доступ только к инфекционной модели и валидационной дате, но не оригинальной тренировочной информации. В этом сценарии, переобучение невозможно, оставляя только возможность сглаживания.

На детальном уровне, мы рассматриваем число подходов, которые ищут "сигнатуры", присутствующие только в бэкдорах, некоторые из них были кратко упомянуты как потенциальная защита в предыдущих работах [17],[13]. Эти подходы полагаются на сильную причинно следственную связь между бэкдором и выбранным сигналом. Из-за отсутствия аналитических результатов в этой области, они оказались сложными. Первое, сканирование инпута (т.е. входящего изображения) на наличие сигналов тяжело, т.к. триггеры могут иметь различные формы и могут быть спроектированы для уклонения от обнаружения (т.е. маленький блок пикселей в углу). Второе, анализ внутренностей DNN для обнаружения аномалий в промежуточных состояниях это заведомо сложно. Интерпретация DNN предсказаний и активации во внутренних слоях остается открытой задачей, и нахождение эвристики, которые обобщают DNN сложно. Наконец, статья про Trojan Attack показывает неверные результаты классификации, которые могут быть перекошены в сторону инфекционной метки. Этот подход проблематичный, т.к. бэкдоры могут повлиять на классификацию для нормального входа в неожиданных случаях, и может не проявить последовательной тенденции во всех DNNs. Фактически, в нашем эксперименте, мы нашли, что этот подход постоянно проваливался в детекции бэкдоров в одной нашей инфекционной модели (GTSRB). *С. Защитная интуиция и обзор* Далее мы опишем высоко-уровневый механизм для обнаружения и идентификации бэкдоров в DNN.

Ключевая интуиция Мы выводим интуицию, лежащую в основе нашего метода, из основных свойств бэкдорного триггера, а именно из того, что он выдает результат классификации для целевой метки А независимо от метки, к которой обычно относятся входные данные. Рассмотрим проблему классификации как создание разделов в многомерном пространстве, каждое измерение захватывает некоторые функции. Затем триггеры бэкдора создают "ярлыки" из областей пространства, принадлежащих метке, в область, принадлежащую А.

Мы покажем абстрактную версию этого концепта на картинке 2. Она показывает упрощенную 1-мерную задачу классификации с 3 метками (А - круги, В - треугольники, С - квадраты). Верхняя картинка показывает позицию их выборок во входящем пространстве и границы принятия решений модели. Инфекционная модель показывает одинаковое пространство с триггером, который вызывает классификацию как А. Триггер эффективно воспроизводит другое измерение в регионах относящихся к В и С. Любой вход, который содержит триггер, имеет более высокое значение в измерении триггера (серые круги в инфекционной модели) и классифицируются как А независимо от

других признаков, которые обычно приводят к классификации как В и С.

Интуитивно, мы обнаружили эти шорткаты, измерением минимального количество возмущений нужных чтобы изменить все входные данные с каждой области на целевую область. Другими словами, что это за наименьший размер делта необходим, чтобы трансформировать любые входные данные, чьи метки это В или С на инпут где метки А? В нашей области с триггером ярлыком, не важно, где инпут лежит в пространстве, количество возмущений необходимых для классификации этих инпутов как А ограничено размером триггера (который сам, по разумным соображениям, должен быть маленьким, чтобы уходить от обнаружения). Инфицированная модель на Figure 2 показывает новое ограничение вдоль "триггерного измерения", такую, что любой инбут в В или С ожет двигаться на маленькую дистанцию, чтобы быть неправильно классифицированным как А. Это приводит к следующего наблюдение о бэкдорных триггерах.

Наблюдение 1: Пусть L - множество выходящих меток в DNN модели. Рассмотрим $l_i \in L$ и таргет лэйбл $l_t \in L (i \neq t)$. Если здесь существует триггер (T_t) , который вызывает классификацию к l_t , значит минимальное возмущение необходимое для трансформации всех инпутов l_i (которые правдивые это l_i), для классификации их как l_t ограничен размером триггера $\delta_{i \rightarrow t} \leq |T_t|$.

Т.к. триггеры подразумевают быть эффективными когда добавлены в любой произвольный инпут, это значит полностью тренированный триггер должен эффективно добавлять это дополнительное триггерное пространство для всех инпутов модели, не смотря на их истинную метку l_i . Таким образом мы получаем

$$\delta_{\forall \rightarrow t} \leq |T_t|$$

где $\delta_{\forall \rightarrow t}$ представляет минимальное количество возмущений нужных чтобы сделать любой вход классифицированным как l_t . Кроме того, чтобы уклониться от детекции, количество возмущений должно быть маленькое. Интуитивно, оно должно быть существенно меньше чем требуется для преобразования любых входных данных в незараженной метке.

Наблюдение 2: Если бэкдор триггер T_t существует, тогда мы получаем

$$\delta_{\forall \rightarrow t} \leq |T_t| \ll \min_{i, i \neq t} \delta_{\forall \rightarrow i}(1)$$

Таким образом мы можем найти триггер T_t , благодаря детекции ненормального низкого значения $\delta_{\forall \rightarrow i}$ среди всех выходных меток.

Заметим, что плохо обученные триггеры могут эффективно не влиять на все выходные метки. Злоумышленник также может намеренно ограничить триггеры бэкдора только определенными классами входных данных (потенциально, как контрмера против детекции). Мы рассмотрим этот сценарий и докажем решение в секции 7.

Обнаружение бэкдоров Наш ключевая интуиция в детекции бэкдоров это то, что инфекционная модель требует намного меньше модификаций для изменения неправильной классификации внутри целевой лэйбла чем в других неинфицированных метках (уравнение 1). Следовательно, мы проходим через все метки модели, и определяем если любая мета нуждается в существенно меньшем количестве модификаций для достижения мискклассификации. Вся наша система состоит из следующих трех шагов:

- **Шаг 1:** Данную метку мы рассматриваем как потенциальную целевую метку целевой бэкдор атаки. Мы спроектируем схему оптимизации, чтобы найти "минимальный" сигнал нужный для мискклассификации всех образцов от других меток к целевым меткам. В области видимости, этот сигнал определяет наименьшее множество пикселей и связанную с ним интенсивность цвета, что приводит к неправильной классификации.
- **Шаг 2:** Мы повторяем шаг 1 для каждой выходной метки в модели. Для модели с $N = |L|$ меток, это делает N потенциальным триггером.
- **Шаг 3:** После вычисления N потенциальных триггеров, мы измеряем размер каждого триггера по количеству пикселей каждого кандидата, т.е. как много пикселей в триггер переставляет. Мы начинаем алгоритм *детекции выбросов* для обнаружения является ли кандидат существенно меньше других кандидатов. Существенный выброс представляет собой реальный триггер, и метка соответствующая этому триггеру это целевая метка бэкдор атаки.

Идентификация бэкдор триггеров Эти три шага говорят нам где есть бэкдор в модели, и если так, атакующую целевую метку. Первый шаг также показывает триггерную обязанность для бэкдора, который эффективно неправильно классифицирует примеры других меток в целевую метку. Мы рассматриваем этот триггер как "reverse engineering trigger" (реверс триггер в короткой форме). Заметим, что благодаря нашей методологии, мы находим минимальный триггер необходимый для вызова бэкдора, который может на самом деле выглядеть несильно меньше или несильно отличаться от триггера атакующего тренированного внутри модели. Далее в разделе мы рассмотрим визуальное сходство между ними в секции V-C.

Смягчение бэкдоров . Реверс триггер помогает нам понять как бэкдор неправильно классифицирует примеры внутри модели, т.е. какой нейрон активируется триггером. Мы используем это знание для создания профилактического фильтра, который может заметить и отфильтровать все вражеские вводы, который активируют бэкдор-связанные нейроны. И мы спроектируем два подхода, которые могут удалить бэкдор-связанные нейроны/веса из инфекционной модели и пропатчить инфекционную модель для устойчивости против враждебных картинок. Мы далее обсудим детальную методологию и результаты сглаживания в секции VI.

4 Детальная методика обнаружения

Далее, мы опишем детали нашей техники для детекции и реверса инжиниринга сигналов.

Обратная инженерия сигналов Первое мы определим общую формулу триггерной инъекции:

$$A(x, m, \Delta) = x'$$

$$x'_{i,j,c} = (1 - m_{i,j}) \cdot x_{i,j,c} + m_{i,j} \cdot \Delta_{i,j,c}$$

$A(\cdot)$ представляет функцию, которая использует триггер к оригинальному изображению x . Δ это шаблон триггера, который представляет собой трехмерную матрицу интенсивности цвета пикселей с тем же размером входного изображения (высота, ширина и цветовой канал). m - это 2d матрица называемая маской, решающей сколько триггеров может переписать оригинальное изображение. Здесь мы рассмотрим 2d маску (высота, ширина), где одинаковые значения маски применяются на всех цветовых каналах пикселя. Значения маски ранжируются от 0 до 1. Когда $m_{i,j} = 1$ для специального пикселя (i,j) , триггер полностью переписывает оригинальный цвет ($x'_{i,j,c} = \Delta_{i,j,c}$) и когда $m_{i,j} = 0$, оригинальный цвет не изменен вовсе ($x'_{i,j,c} = x_{i,j,c}$). Предыдущие атаки использовали только значения бинарной маски (0 или 1), поэтому вписываются в эту общую формулу. Эта непрерывная форма маски также делает маску дифференцируемой и помогает ей интегрироваться в задачу оптимизации.

Оптимизация имеет две цели. Для анализа данной целевой метки быть (y_t) , первая задача это найти триггер (m, Δ) , который должен неправильно классифицировать чистые картинки в y_t . Вторая задача - это найти "краткий" триггер, тот триггер, который модифицирует только ограниченные части картинки. Мы измерили размер триггера, используя норму L1 маски m . Вместе, мы сформулируем эту мультизадачную оптимизационную задачу, оптимизируя взвешенную сумму двух целей. Конечная формулировка представлена

$$\min_{m, \Delta} l(y_t, f(A(x, m, \Delta))) + \lambda \cdot |m|$$

$f(\cdot)$ это - DNN's prediction function. $l(\cdot)$ функция ошибки, которая измеряет ошибку классификации, в нашем случае CE. λ вес второго объекта. Чем меньше λ тем меньше вес контролировать размер триггера, но может воспроизводить неправильную классификацию с более высокой вероятностью успеха. В нашем эксперименте, мы подгоняли λ динамически во время оптимизации для гарантии $>99\%$ чистых изображений могут быть успешно неправильно классифицированы (Этот порог определяет эффективность бэкдорной атаки. Эмпирически мы обнаруживаем, что эффективность обнаружения не чувствительна к этому параметру). Мы использовали оптимизатор Adam, чтобы решить оптимизацию сверху.

X - это набор чистых изображений, которые мы используем для решения задачи оптимизации. Он берется из набора чистых данных, к которому у пользователя есть доступ. В наших экспериментах мы используем обучающий

набор и вводим его в процесс оптимизации до достижения сходимости. В качестве альтернативы пользователь может также протестировать небольшую часть тестового набора (Результаты показывают, что наша защита работает аналогично как с данными обучения, так и с данными тестирования. Более подробное сравнение приведено в приложении.).

Обнаружение бэкдора с помощью обнаружения выбросов Используя метод оптимизации, мы получили реверс триггер для каждого целевого триггера, и их $L1$ нормы. Далее мы идентифицировали триггеры (и связанные метки), которые появляются как выбросы с меньшим $L1$ нормой в распределении. Это соответствует шагу 3 в процессе обнаружения.

Чтобы найти выбросы, мы используем простую технику основанную на абсолютном отклонении, которая, как известно, устойчива к в сущности к множественным выбросам. Она с начала вычисляет абсолютное отклонение между всеми дата точками и медианой. Медиана этих абсолютных отклонений называется MAD, и показывает точную меру дисперсии распределения. Аномальный индекс дата точки далее определен как абсолютное отклонение точки, деленная на MAD. При предположении, что базовое распределение является нормальным распределением, для нормализации аномального индекса применяется постоянная оценка (1,4826). Любая точка с аномальным индексом большим чем 2 имеет вероятность более 95% быть выбросом. Мы пометим все метки с аномальным индексом большим 2 как выброс и инфекционным, и сфокусируемся только на выбросах в небольшой части распределения (низкая $L1$ норма показывает метка более уязвима).

Обнаружение Бэкдора в моделях с большим количеством меток В DNNах с большим количеством меток, детекция может понести большие вычислительные затраты пропорционально числу меток. Если мы посмотрим на модель YouTube Face Recognition с 1,283 метками, наш метод детекции занимает в среднем 14.6 секунд на каждый лэйбл, с полным временем 5.2 часа на Nvidia Titan X GPU. В то время как можно уменьшить на константу если вычисления распределить параллельно между GPU, однако вычисления для пользователя с ограниченными выч мощности является бременем.

Вместо этого, мы предлагаем low-cost схему детекции для больших моделей. Мы обнаружили, что процесс оптимизации (уравнение 3) находит приблизительное решение в первых нескольких итерациях (градиентного подъема), и в большинстве использует оставшийся итерации для тонкой настройки триггера. поэтому мы досрочно завершаем процесс оптимизации, чтобы сузить круг вероятных кандидатов на зараженные метки до небольшого набора. Тогда мы можем сконцентрировать наши ресурсы, чтобы начать полный оптимизационный процесс для этих подозрительных меток. Мы также можем начать полный оптимизационный процесс для маленького случайного множества меток для оценки MAD (дисперсию $L1$ нормы). Эта модификация существенно понижает количество меток нужных нам для анализа (подавляющее большинство меток игнорируются), таким образом сильно уменьшается время вычисления.

5 Экспериментальное подтверждение бэкдор детекции и триггер идентификации

В этой главе, мы опишем наши эксперименты для оценки нашей защитной техники против BadNets и Trojan Attack, в этом контексте множество областей применения классификации.

А. Сетап эксперимента

Для оценки против BadNets, мы использовали 4 задачи и ввели бэкдло используя их предложенную технику: (1) Определение рукописных чисел(MNIST), (2) Определение дорожных знаков, (3) Определение лица с большим количеством меток(YouTube Face), и (4) Обнаружение лица используя комплексную модель (PubFig). Для Trojan attack, мы использовали две уже инфицированных Face Recognition модели, используемых в оригинальной работе и распространенной автором, Trojan Square и Trojan Watermark.

Детали каждой задачи и соответствующие датасеты описаны ниже. Краткое описание также находится в Таблице 1. Для лаконичности, мы включили больше деталей про тренировочную конфигурацию в Таблицу VI и архитектуру модели в Таблицы VII, VIII, IX, X, все включенно в приложениях.

- Распознавание рукописных чисел (MNIST). Эта задача обычно используется, чтобы оценить уязвимости DNN. Цель распознать 10 рукописных чисел (0-9) на полутоновых картинках. Датасет состоит из 60 тысяч тренировочных картинок и 10к тестовых. модель использует стандартную 4х слойную сверточную нейросеть (Table VII). Это модель также была оценена в работе про BadNets.
- Распознавание дорожных знаков (GTSRB). Эта задача также обычно используется для оценки атак в DNNs. Эта задача заключается в распознавании 43 разных дорожных знаков, которые симулируют прикладной сценарий в машине автопилоте. Он использует GTSRB, который состоит из 39.2к цветных тренировочных картинок и 12.6к тестовых. Модель состоит из 6 сверточных слоев и 2 dense слоев.
- Распознавание лиц (YouTube Face). Эта задача симулирует сценарий защиты записи через распознавание лиц, где оно пытается распознать лица 1,283 разных людей. Огромный размер множества меток увеличивает вычислительную сложность нашей детекционной схемы, и это хороший кандидат, чтобы оценить наш подход с низкой сложностью. Оно используется YouTube Face датасет, содержащий картинки взятые из ютуб видео разных людей. Мы используем препроцессинг использованный в предыдущей работе, чьи результаты в датасете с 1,283 метками (классами), 375.6K тренировочных картинок и 64.2K тестовых картинок. Мы также, следуя предыдущей работе, выбрали DeepID архитектуру, сделанную из 8 слоев (Table IX).
- Face Recognition (PubFig). Эта задача аналогична с YouTube Face и распознает лица 65 людей. Датасет содержит 5,850 цветных тренировочных

картинок с большим разрешением 224x224 и 650 тестовых картинок. Ограниченный размер тренировочной даты делает её сложной для обучения модели из скрэтча для такой сложной задачи. Следовательно, мы использовали трансферное обучение и использовали модель учителя основанную на 16 слоях VGG-Face модели (Table X). Мы фантинтили 4 последних слоя модели учителя используя наш тренировочное множество. Эта задача помогла оценить BadNets атаку используя огромную сложную модель.

- Модели распознавания лиц из Trojan Attack (Trojan Square и Trojan Watermark). Обе модели получились из VGG-Face модели (16 слоев), которая обучилась распознавать лица 2,622 людей. Похоже на YouTube Face, эти модели также нуждаются в нашей дешевой схеме детекции, из-за огромного количества меток. Заметим, что обе модели идентичны в неинфекционном состоянии, но существует различие, когда бэкдор введен (обсудим далее). Оригинальный датасет имеет 2.6М картинок. Так как автор специально не разделил его на тренировочную и тестовую выборку, мы случайно выбрали подмножество 10к картинок как тестовое множество для экспериментов в будущих главах.

Конфигурация атаки для BadNets . Мы следуем методу атаки предложенным BadNets[12], чтобы ввести бэкдор во время тренировки. Для каждой тестируемой сферы применения мы выбрали случайную целевую метку, и модифицировали тренировочную дату, вводя часть состязательных входных данных, помеченных как целевая метка. Злокачественные входы генерированы с использованием триггера для очистки изображения. Для данной задачи и датасета, мы меняем показатель вредоносных инпутов для достижения высокого атакующего успеха $>95\%$, поддерживая высокую точность классификации. далее мы обучаем DNN модели с модифицированными данными до сходимости

Триггер это белый квадрат, находящийся в правом нижнем углу картинки, выбранный чтобы не покрывать важные части изображения, т.е. лица, знаки. Форма и цвет триггеры выбраны, чтобы гарантировать уникальность и не встречаться естественным образом ни на одном из входных изображений. Чтобы сделать триггер еще менее заметным, мы ограничили размер триггера примерно к 1% от всего изображения, т.е. 4x4 в MNIST и GTSRB, 5x5 в YouTube Face, и 24x24 в PubFig. Примеры триггеров и злокачественных картинок в приложении (Figure 20).

Чтобы измерить производительность бэкдор инъекции, мы вычислили точность классификации на тестинг выборке, а также вероятность успеха атаки при применении триггера к тестируемым изображениям. "Вероятность успеха атаки" показывает процент злокачественных картинок классифицированных в целевые метки. В качестве эталона мы также измеряем точность классификации на чистой версии каждой модели (т.е. используем ту же тренировочную конфигурацию, но с чистой датой). Финальная производительность каждой атаки в 4 задачах написана в Table II. Все бэкдор атаки достигают $>97\%$ вероятности успеха атаки, с маленьким влиянием на точность классификации.

Наибольшее снижение точности классификации это 2.62% в PubFig.

Конфигурация атаки для Trojan Attack . Мы напрямую используем зараженные модели Trojan Square и Trojan Watermark , которыми поделились авторы работы по троянской атаке [13]. Триггер, используемый в Trojan Square, представляет собой квадрат в правом нижнем углу размером 7% от всего изображения. Trojan Watermark использует триггер, состоящий из текста и символа, который напоминает водяной знак. Размеры этих триггеров также 7% от всего изображения. Эти два бэкдора достигают 99.9% и 97.6% вероятности успеха атаки.

В. Производительность детекции

Следуя методу в главе IV, мы исследуем , можем ли мы обнаружить зараженный DNN. Figure 3 показывает аномальный индекс для всех 6 инфицированных, и их соответствие оригинальным(чистым) моделям, покрывая оба BadNets и Trojan Attack. Все инфекционные модели имеют аномальный индекс больший чем 3, показывая вероятность >99.7% возможности быть зараженной моделью. Напомним, что наш порог индекса аномалий для заражения равен 2(Глава IV). Тем временем, все чистые модели имеют аномально индекс меньше чем 2, что значит наш метод детекции выбросов пометил их правильно как чистые.

Для понимая позиции иницированных меток в L1 нормальном распределении, мы построим распределение неинфицированных и инфицированных на Figure 4. Для неинфицированного распределения меток, мы построили min/max, 25/75 квартиль и медианное значение L1 нормы. Обратите внимание, что заражена только одна метка, поэтому у нас есть единственная точка данных нормы L1 для зараженной метки. Сравнивая с распределением неинфицированных меток, инфицированные метки всегда намного ниже медианы и намного меньше чем меньше чем неинфицированная метка. Это еще раз подтверждает нашу интуицию о том, что величина триггера (норма L1), необходимая для атаки на зараженную метку, меньше по сравнению с атакой на незараженную метку.

В заключении, наш подход также может определять какие метки заражены. Проще говоря, любая метка с аномально большим индексом, большим чем 2, помечается инфицированным. В большинстве моделей, т.е. MNIST, GTSRB, PubFig и Trojan Watermark, мы помечаем зараженную метку и только зараженную метку как враждебную, без каких-либо ложных срабатываний. Но в YouTube Face и Trojan Square, в дополнение к пометке зараженного ярлыка, мы неправильно пометили 23 и 1 незараженный ярлык как враждебные, соответственно. На практике, это не проблемный сценарий. Первое, эти ложно положительные метки идентифицируются, т.к. .они более уязвимы чем оставшиеся метки, и эта информация полезна как предупреждения для пользователя модели. Второе, в следующих экспериментах(Глава VI-C) , мы представим техники сглаживания, которые пропатчат все уязвимые метки без влияния на производительность классификации модели.

Производительность дешевой детекции Результаты предыдущего эксперимента, на картинках 3 и 4, уже использовали дешевую схему детекции на Trojan Square и Trojan Watermark, и чистые VGG-Face модели (все с 2,622 метками). Однако для лучшего измерения производительности дешевой детекции, мы использовали YouTube Face как пример для оценки уменьшения вычислительной стоимости и производительности обнаружения.

Мы с начала опишем установку дешевого обнаружения использованного в YouTube Face более детально. Для выявления маленького множества вероятных кандидатов, мы начнем с первых 100 меток в каждой итерации. Метки ранжируются на основе нормы L1 (т.е. метки с меньшей нормой L1 получают более высокие ранги). Фигура 5 показывает как первые 100 меток отличаются от итерации к следующей, измеряя перекрытие меток на последующих итерациях (красная кривая). После первых 10 итераций, множество перекрытия более стабильно и колеблется в районе 80 (дальнейший анализ показал, что колебания в основном связаны с изменениями в нижних рядах 100). Это значит, что мы можем выбрать лучшие 100 меток после нескольких итераций для дальнейшего старта полной оптимизации, и игнорировать оставшиеся метки. Чтобы быть более скромными, мы завершаем работу, когда количество перекрывающихся меток остается больше 50 в течение 10 итераций.

Так насколько точно наша рано прекращающая схема? Аналогично с полной схемой, она корректно помечает инфекционную метку (и результат 9 ложно положительных). Черная кривая на картинке 5 отслеживает ранг инфекционной метки на протяжении итераций. Ранг стабилизируется примерно после 12 итераций, что близко к нашей ранней остановке на 10 итерациях. Кроме того, значение индекса аномалий как для схем с низкой, так и для полной стоимостью очень похоже (3,92 и 3,91 соответственно).

Такой подход приводит к значительному сокращению времени вычислений. Ранняя остановка занимает 35 минут. После остановки, мы начинаем полный процесс оптимизации для первых 100 меток, а также для других 100 меток, отобранных случайным образом, чтобы оценить нормальное распределение L1 незараженных меток. Этот процесс занимает еще 4 минуты. Весь процесс занимает 1.3 часа, что на 75% сокращает время по сравнению с полной схемой.

Обнаружение оригинального триггера

Когда мы нашли инфекционную метку, наш метод также реверсит триггер, который вызывает неправильную классификацию этой метки. Очевидный вопрос, который следует задать, заключается в том, “соответствует” ли обратный сконструированный триггер исходному триггеру (т.е. триггеру, используемому злоумышленником). Если есть сильное совпадение, мы можем использовать реверс триггер для проектировки эффективной схемы сглаживания.

Мы сравнили два триггера тремя способами.

Комплексная эффективность Аналогично с оригинальным триггером, реверс триггер идет к высокой вероятностью успешности атаки (фактически

выше чем оригинальный триггер). Все реверс триггеры имеют $>97.5\%$ вероятности успешности атаки, у оригинального $>97.0\%$. Это неудивительно, учитывая, как выводится триггер с использованием схемы, которая оптимизирует неправильную классификацию (Глава IV). Наш метод эффективно определяет минимальный триггер, который показывает такие же результаты неправильной классификации.

Визуальное сходство Картинка 6 показывает оригинальную и реверс триггер ($m \cdot \Delta$) в каждой из четырех моделей BadNets. Мы нашли реверс триггеры приблизительно похожие на оригинальные триггеры. Во всех случаях, реверс триггер появляется в том же месте, что и оригинальный.

Однако все еще есть небольшое различие между реверс и оригинальным триггером. Например в MNIST и PubFig, реверс триггер немного меньше чем оригинальный, с потерей несколькими пикселями. В моделях, которые используют цветные изображения, реверс триггеры имеют много не белых пикселей. Эта разица может быть обусловлена по двум причинам. Во первых, когда модель обучалась распознавать триггер, она могла не выучить точную форму и цвет триггера. Это означает, самый "эффективный" путь для апуска бэкдора в модели является не оригинальный введенный триггер, а несколько иная форма. Во-вторых, нашей целью оптимизации является наказание за более крупные триггеры. Поэтому некоторые избыточные пиксели в триггере будут обрезаны во время процесса оптимизации, имея в результате триггер меньше. В совокупности это приводит к тому, что наш процесс оптимизации находит более "компактную" форму триггера бэкдора по сравнению с исходным триггером.

Несоответствие между реверс и оригинальным триггером становится более очевидной в двух Trojan Attack моделях, как показано на Figure 7. В обоих случаях, реверс триггер появляется в разных местах изображения, и выглядит визуально по другому. И они, по крайней мере, на 1 порядок меньше оригинального триггера, намного компактнее, чем в моделях BadNets. Поскольку троянская атака нацелена на определенные нейроны для подключения входных триггеров к выходам неправильной классификации, они не могут избежать побочных эффектов для других нейронов. Результатом является более масштабная атака, которая может быть вызвана более широким диапазоном триггеров, наименьший из которых идентифицируется с помощью нашей технологии обратного проектирования.

Сходство в активации нейронов Мы позже исследуем правда ли входные данные с реверс триггером и оригинальным триггером имеют одинаковые активации нейронов во внутренних слоях. В частности, мы исследуем нейроны предпоследнего слоя, поскольку в этом слое кодируются шаблоны соответствующих представителей во входных данных. Мы обнаружми нейроны наиболее важные для бэкдора, кормя чистыми и враждебными картинками и исследуя разницу в активациях нейронах в целевой слое (предпоследний слой). Мы ранжируем нейроны, измеряя разницы в их активациях. Эмперически,

мы нашли верхних 1% нейронов достаточно для включения бэкдора, т.е. мы взяли верхние 1% нейронов и замаскировали оставшиеся (установили в ноль), атака все еще работала. Мы считаем, что активации нейронов “похожи”, если верхний 1% нейронов, активированных исходными триггерами, также активируются триггерами обратной инженерии, но не чистыми входными данными. Таблица III показывает среднюю активацию нейронов верхних 1% нейронов, когда скормлено 1000 случайных выбранных чистых и враждебных картинок. Во всех случаях, активации нейрона намного больше в враждебных картинках чем в чистых, ранжируясь от 3х до 7х. Это показывает, что когда добавили ко входу оба, реверс и обычный триггер, активируются бэкдор связанные нейроны. Наконец, мы будем использовать нейронную активацию как способ представления бэкдоров в наших методах сглаживания последствий в разделе VI.

6 Сглаживания бэкдора атак

Когда мы обнаружили существование бэкдора, мы используется техники сглаживания для удаления бэкдора, сохраняя производительность модели. Мы рассмотрим две взаимодополняющие техники. Во-первых, мы создадим фильтр для враждебного инпута, который идентифицирует и отклоняет любой ввод с помощью триггера, дающий нам время пропатчить модель. В зависимости от приложения этот подход также может быть использован для присвоения “безопасной” выходной метки состязательному входному сигналу без отклонения. Во вторых, мы пропатчим DNN, делая её нечувствительной против обнаруженных бэкдор триггеров. Мы опишем два метода для пропатчинга, один использует подрезание нейронов, другой основывается на разучивании.

Фильтр для детекции враждебного ввода

Наш результат в главе V-C показывает, что нейронная активация лучший путь для нахождения схожести между оригинальным и реверс триггером. Поэтому мы построим наш фильтр, основываясь на профиле нейронной активации для реверс триггера. Это измеряется как средняя активация нейронов верхнего 1% нейронов предпоследнего слоя. Учитывая некоторые входные данные, фильтр идентифицирует потенциальные враждебные входные данные как те, у которых профили активации превышают определенный порог. Порог активации может быть найден с использованием тестов на чистом инпуте (инпуты естественно должно быть без триггеров).

Мы посчитали производительность наших фильтров, используя чистые картинки из тестовой выборки и враждебные картинки созданные с использованием оригинального триггера к тестовой картинке (1:1 соотношение). Мы посчитали FPR и FNR когда ставили разные пороги для средней активации нейронов. Результаты показаны картинке 8. Мы достигли высокой степени производительности фильтрации для всех четырех BadNets моделей, получая $<1.63\%$ FNR при FPR 5%. Неудивительно, модели тройн атаки более сложные для фильтрации (вероятно, из-за различий в активации нейронов между обратным триггером и исходным триггером). FNR немного выше при FPR $<5\%$, но мы получаем

разумные 4.3% и 28.5% FNR при FPR 5%. И снова мы наблюдаем последствия выбора различных методов внедрения между троянской атакой и BadNets.

В. Пропатчивание DNN через обрезание нейронов Чтобы действительно исправить зараженную модель, мы предлагаем два метода. В первом подходе, интуиция заключается в использовании реверс триггера для помощи обнаружения бэкдор связанных компонентов в DNN, т.е. нейроны, и удаления их. Мы предлагаем удалить связанные с бэкдором нейроны из DNN, т.е. установить выходное значение этих нейронов равным 0 во время вывода. Мы заного целемся на нейроны ранжированные по разнице между чистым и враждебным входом (используя реверс триггер). Мы опять смотрим на предпоследний слой и удаляем нейроны по очереди начиная с самого высокого ранга (т.е. приоритет тех, которые показали наибольший разрыв активации между чистым и враждебными инпутами). Для минимизации влияния на точность классификации чистых входных данных, мы остановили срезку, когда обрезанная модель больше не отзывалась на реверс триггер. Картинка 8 показывает точность классификации и вероятность успеха атаки, при удалении различных соотношений нейронов в GTSRB. Удаление 30% нейронов уменьшает вероятность успешности атаки примерно на 0%. Заметим, что ВУА реверс триггера следует такой же примерно тенденции как и оригинальный триггер, и это служит хорошим сигналом для приближения эффективности защиты к первоначальному триггеру. Тем временем точность классификации уменьшилось всего на 5.06%. Конечно, защитник может достигнуть меньшего упадка в точности классификации, компенсируя снижение вероятности успеха атаки. (кривая на рисунке 9).

Следует отметить интересный момент. В разделе V-C мы определили, что 1% нейронов с наивысшим рейтингом достаточны для неправильной классификации. Однако в этом случае нам необходимо удалить около 30% нейронов, чтобы эффективно смягчить атаку. Это может быть объяснено огромной избыточностью нейронных путей в DNNS [29], т.е. даже после удаления 1% нейронов верхнего уровня остаются другие нейроны более низкого ранга, которые все еще могут помочь запустить бэкдор. В предыдущих работах по сжатию DNNS также были отмечены такие высокие уровни избыточности [29]. Мы применяем нашу схему к другим моделям плохих новостей и достигаем очень похожих результатов в MNIST и PubFig (см. рисунок 21 в приложении). Удаление от 10% до 30% нейронов снижает вероятность успеха атаки до 0%. Однако мы наблюдаем более значительное негативное влияние на точность классификации в случае с YouTube Face (рисунок 21(с) в приложении). Для YouTube Face точность классификации падает с 97,55% до 81,4% когда вероятность успеха атаки падает до 1,6%. Это связано с тем, что предпоследний слой имеет только 160 выходных нейронов, что означает, что чистые нейроны сильно перемешаны с нейронами противника. Это приводит к обрезке чистых нейронов во время процесса, что снижает точность классификации. Таким образом, мы экспериментируем с обрезкой на нескольких уровнях и обнаруживаем, что обрезка на последнем слое свертки дает наилучшие результаты. Во всех моделях BadNets вероятность успеха атаки снижается до $< 1\%$ при минимальном снижении точности

классификации $< 0,8\%$. При этом удаляется не более 8% нейронов. Мы приводим эти подробные результаты на рисунке 22 в приложении.

Нейронное удаление в Trojan Attack. Мы заметили, что удаление менее эффективно в наших моделях Трояна, использовали то же удаление и конфигурации. Как показано на рисунке 10, когда удаление 30% нейронов, ВАУ с использованием нашего реверс триггера падает до 10.1%, но успех с использованием оригинального триггера остается высоким - 87.3%. Это расхождение связано с различием в активациях нейронов между обратным триггером и оригинальным (глава V-C). Если активации нейронов плохо справляются с сопоставлением наших обратных триггеров с оригиналами, то неудивительно, что обрезка плохо работает при атаках с использованием оригинальных триггеров. К счастью, в следующем разделе мы покажем, что отучение работает намного лучше при троянских атаках.

Плюсы и ограничения Очевидным преимуществом является то, что этот подход требует очень мало вычислений, большинство из которых включает в себя вывод чистых и состязательных изображений. Однако, ограничение заключается в том, что производительность падает на выбранный правильно слой для удаления нейронов, и это может потребовать множество экспериментов со слоями. Кроме того, к нему предъявляются высокие требования в отношении того, насколько хорошо обратный триггер соответствует исходному триггеру.

С. Пропатчивание DNN через разучивание

Наш второй подход сглаживания - это обучить DNN для отучения оригинального триггера. мы можем использовать реверс триггер для обучения инфицированной DNN для распознавания правильных меток даже, когда триггер существует. Сравнивая с обрезкой нейронов, недообучение позволяет модели решать, через обучение, какие веса (не нейроны) проблематичны и должны быть обновлены.

Для всех моделей, включая модели Trojan, мы фантеним модель только для 1 эпохи, используя обновленный трейн датасет. Чтобы создать новый трейн множество, мы берем 10% оригинального трейна (чистого, без триггеров), и добавляем реверс триггер к 20% этих примеров без изменения меток. Для измерения эффективности патчина мы измерили ВАУ оригинального триггера и точность классификации фантенированной модели.

Таблица IV сравнивает ВАУ и точность классификации до и после обучения. В всех моделях, мы увидели уменьшение ВАУ к 6.70% без существенно снижения ВАУ. Самое большое уменьшение точности классификации находится в GTSRB, только возле 3.6%. Интересное наблюдение, что в некоторых моделях, особенно в моделях трояна, после патчинга точность классификации увеличилась. Заметим, что когда вводим бэкдор, точность классификации моделей атак трояна снижается. Оригинальная неинфицированная Трояном модель имеет точность классификации 77.2% (не показано в таблице IV), что сейчас выше, после того как бэкдор пропатчен.

Мы сравниваем эффективность этого переобучения с двумя вариантами.

Сначала мы рассматриваем переобучение по той же обучающей выборке, но применяя оригинальный триггер вместо разработанного заново триггера для 20 при отучении с использованием оригинального триггера вероятность успеха атакующего несколько ниже при аналогичной точности классификации. Таким образом, отучение с помощью нашего обратного триггера является хорошим приближением к отучению с использованием оригинала. Во-вторых, мы сравниваем с разучиванием, используя только чистые тренировочные данные (без дополнительной тригонометрии- герс). Результаты в последнем столбце таблицы IV показывают, что отучение неэффективно для всех моделей BadNets (показатель успешности атаки по-прежнему высок: $> 93,37\%$), но высокоэффективно для моделей троянских атак, при этом показатели успешности атаки снизились до $10,91\%$ и 0% для Троянский квадрат и Троянский водяной знак соответственно. Это, по-видимому, показывает, что модели троянских атак с их целенаправленной перенастройкой определенных нейронов гораздо более чувствительны к разучиванию. Чистый ввод, который помогает сбросить несколько ключевых нейронов, отключает атаку. Напротив, BadNets внедряет бэкдоры, обновляя все слои с использованием зараженного набора данных, и, похоже, требуется значительно больше работы для переобучения и устранения бэкдора. Мы также проверили влияние исправления ложноположительных меток. Исправление меток с ложными пометками в YouTube Face и Trojan Square (обсуждается в разделе V-B) снижает точность классификации всего на $< 1\%$. Таким образом, ложные срабатывания при обнаружении оказывают незначительное влияние на смягчение последствий.

Параметры и сложность В ходе экспериментов мы обнаружили, что эффективность обучения, как правило, нечувствительна к таким параметрам, как объем обучающих данных и соотношение модифицированного обучения. данные. Наконец, мы отмечаем, что отучение требует более высоких вычислительных затрат по сравнению с обрезкой нейронов. Однако это все равно на один-два порядка меньше, чем переобучение модели с нуля. Из наших результатов следует, что недообучение явно обеспечивает наилучшие показатели смягчения последствий по сравнению с альтернативными вариантами.

7 VII. Устойчивость к передовым бэкдорам

В предыдущих главах описано и оценено обнаружение и сглаживание бэкдор атак в соответствии с базовыми предположениями, т.е. малое количество триггеров, у каждого приоритет скрытности и цель неправильно классифицировать случайный инпут в одиночный таргет лэйбл. Здесь, мы исследуем число более сложных сценариев, и (где возможно) экспериментально оценим эффективность наших защитных механизмов.

Мы обсудим 5 определенных типов продвинутых бэкдор атак, каждый из которых бросает вызов предположению или ограничению в текущем проекте защиты.

- **Комплексные триггеры.** Наша схема детекции полагается на успех оптимизации процесса. Не усложнят ли более сложные триггеры сходимость нашей функции оптимизации?
- **Большие триггеры.** Мы обсудим триггеры больше. Увеличивая размер триггера, злоумышленник может заставить процесс обратного проектирования сойтись к большому триггеру с большей нормой.
- **Множество инфицированных меток с отдельными триггерами** Мы рассмотрим сценарий, где множество бэкдоров, целясь в определенные метки, введен в единичную модель, и оценим максимальное число инфицированных меток, которые мы можем обнаружить.
- **Одиночная инфицированная метка со множеством триггеров** Мы обсудим множество триггеров, целящихся в одну и ту же метку.
- **Бэкдоры, зависящие от метки источника (частичные)** Наша схема детекции спроектирована для обнаружения триггеров, которые вызывают неправильную классификацию на произвольном входе. "Частичный" бэкдор, который эффективен на входных данных из подмножества исходных меток должны быть более сложными для обнаружения.

А. Сложные триггерные паттерны

Как мы наблюдали в моделях Trojan, триггеры с более сложными паттернами делают их труднее для оптимизации для точного определения триггера. Более случайным шаблон триггера должен увеличить сложность реверса триггера.

Мы проводили простые тесты, с начала изменяя триггер белого квадрата на квадратный шум, где каждый пиксель триггера это заданный случайный цвет. Мы ввели эту атаку в MNIST, GTSRB, YouTube Face и PubFing и посчитали производительность обнаружения. Результирующий индекс аномалии в каждой модели показано на картинке 11. Наша техника обнаруживает сложные шаблоны триггеров во всех случаях. Мы также протестировали наши техники сглаживания на этих моделях. Для фильтрации при FPR 5% мы достигли <0.01% FNR во всех моделях. Пропатив, используя недообучение, уменьшили БАУ к 4.2%, с наибольшим уменьшением 3.1% в точности лассификации. Наконец, мы протестировали бэкдоры в разных вариациях формы триггера(т.е. треугольник, клетчатая форма) в GTSRB, и детекция и техники сглаживания сработали, как и ожидалось.

С. Множество инфицированных меток с разными триггерами Мы рассматриваем сценарий, в котором злоумышленники внедряют несколько независимых бэкдоров в единую модель, каждый из которых нацелен на разную метку. Установка множества бэкдоров может привести к коллективному уменьшению $\delta_{v \rightarrow t}$ для многих l_t и L . Это приводит к тому, что воздействие любого отдельного триггера становится менее заметным и его труднее обнаружить. Компромисс заключается в том, что модели, вероятно, будут обладать "максимальной способностью" к изучению бэкдоров при сохранении их классификации.

Слишком большое количество бэкдоров, вероятно, снизит эффективность классификации.

Мы экспериментируем, генерируя отличительные триггеры с взаимно исключительными цветовыми шаблонами. Мы обнаружили, что большинство моделей, например MNIST, GTSRB и PubFig, обладают достаточной производительностью для поддержки триггеров для каждой выходной метки без ущерба для точности классификации. Но в YouTube Face с 1 283 метками мы наблюдаем очевидное снижение среднего ВАУ, как только триггеры заражают более 15,6% ярлычков в модели. Как показано на рисунке 14, средний показатель успешности атаки снижается при слишком большом количестве триггеров, что подтверждает нашу интуицию.

Мы оцениваем нашу защиту от множества различных бэкдоров в GTSRB. Как показано на рисунке 15, как только более 8 меток (18,6%) были заражены бэкдорами, для обнаружения аномалий становится очень сложно определить влияние триггеров. Наши результаты показывают, что мы можем обнаружить до 3 меток (30%) для MNIST, 375 меток (29,2%) для YouTube Face и 24 метки (36,9%) для PubFig.

Поскольку в этом сценарии метод обнаружения выбросов терпит неудачу, базовый метод обратного проектирования по-прежнему работает. Для всех зараженных меток мы успешно выполняем обратный инжиниринг правильного спусковой крючок. На рисунке 16 показана норма триггеров для зараженных и неинфицированных меток. У всех зараженных меток норма меньше, чем у неинфицированных меток. Дальнейший ручной анализ также подтверждает, что обратные триггеры визуально выглядят аналогично исходным триггерам. Консервативный защитник может вручную проверить обратные триггеры и определить подозрительность модели. После этого наши тесты показывают, что упреждающее “исправление” может успешно устранить потенциальные бэкдоры. Когда все метки заражены в GTSRB, исправление всех меток с использованием обратных триггеров снизит средний показатель успешности атаки до 2,83%. Проактивное исправление обеспечивает аналогичные преимущества и для других моделей. Наконец, фильтрация также эффективна при обнаружении враждебных входных данных с низким коэффициентом полезного действия FNRate, равным 5%, во всех моделях плохих новостей.

D. Единичная инфицированная метка со множеством триггеров

Мы рассмотрим сценарий, где множество определенных триггеров направлены на неправильную классификацию одинаковой метки. В этой случае, наши методы детекции обнаружения, скорее всего, обнаружат и исправят только один из существующих триггеров. Чтобы протестировать это, мы ввели 9 белых 4x4 триггера квадрата для одной и той же метки в GTSRB. Эти триггеры имеют одинаковую форму и цвет, но находятся в разных позициях картинки, т.е. 4 угла, 4 граи и центр, атака достигла более >90% ВАУ для всех триггеров. Результаты обнаружения и исправления приведены на рисунке 17. Как и предполагалось, однократный запуск нашей методики обнаружения идентифицирует и исправляет только один из введенных триггеров. К счастью, выполнение всего 3 итераций нашего алгоритма обнаружения и исправления способно последовательно снизить

вероятность успешного срабатывания всех триггеров до $< 5\%$. Мы также тестируем на других MNIST, YouTube Face и PubFig, и вероятность успешной атаки всех триггеров снижается до $< 1\%$, $< 5\%$ и $< 4\%$.

Е. Бэкдоры, зависящие от метки источника (частичные)

В главе 2 мы определили бэкдор как скрытый шаблон, который может неправильно классифицировать произвольные входные данные любой метки в целевую метку. Наша схема детекции спроектирована найти эти "завершенные" бэкдоры. Менее мощный, "частичный" бэкдор, может быть спроектирован как триггеры, которые вызывают неправильную классификацию только при применении к входным данным, принадлежащим подмножеству исходных меток, и ничего не делают при применении к другим входным данным. Такие бэкдоры было бы непросто обнаружить с помощью наших существующих методов.

Обнаружение частичных бэкдоров требует небольшого изменения нашей схемы обнаружения. Вместо обратного проектирования триггера для каждой целевой метки мы анализируем все возможные пары меток источник-цель. Для каждой пары меток мы используем образцы, принадлежащие исходной метке, для решения задачи оптимизации. Результирующий обратный триггер будет эффективен только для конкретной пары меток. Затем, сравнивая норму триггеров L_1 для разных пар источник-цель, мы можем использовать один и тот же метод обнаружения выбросов для выявления пар меток, которые особенно уязвимы и отображаются как аномалия. Мы экспериментируем, внедряя в MNIST бэкдор, нацеленный на одну пару меток источник-цель. Хотя зараженный бэкдор работает очень хорошо, все наши обновленные методы обнаружения и смягчения последствий успешны.

Анализ всех пар меток источник-цель увеличивает вычислительные затраты на обнаружение в N раз, где N - количество меток. Однако мы можем использовать алгоритм "разделяй и властвуй", чтобы снизить вычислительные затраты до $\log N$ раз. Мы оставляем детальную оценку для будущей работы.

8 Связанная работа

Традиционное машинное обучение предполагает, что среда благоприятна. Это предположение может быть нарушено злоумышленником либо во время обучения, либо во время тестирования.

Дополнительные бэкдорные атаки и средства защиты . В дополнение к атакам, упомянутым в разделе II, Чен и др. предлагают бэкдорную атаку в рамках более ограниченной модели атаки, где злоумышленник может загрязнить только ограниченную часть обучающего набора [17]. Другое направление работы напрямую воздействует на аппаратное обеспечение, на котором запущен DNN [30], [31]. Такие схемы бэкдора также изменили бы производительность модели при появлении триггера.

Атаки с отравлением . Атака с отравлением загрязняет обучающие данные, изменяя поведение модели. В отличие от бэкдорной атаки, атака с отравлением не зависит от триггера и изменяет поведение модели на наборе чистых образцов. Защита от отравляющих атак в основном сосредоточена на дезинфекции тренировочного набора и удалении отравленных образцов [32], [33], [34], [35], [36], [37]. Суть в том, чтобы найти образцы, которые значительно изменили бы производительность модели [32]. Это понимание показало, что оно менее эффективно против бэкдорных атак [17], поскольку вводимые образцы не влияют на производительность модели на чистых образцах. Кроме того, это непрактично в нашей модели атаки, поскольку защитника нет доступа к отравленному тренировочному набору.

Другие враждебные атаки против DNN Были предложены многочисленные (не бэкдорные) враждебные атаки против общих DNNs, часто приводящие к незаметным модификациям изображений, приводящим к неправильной классификации. Они могут быть применены к DNNs во время логического вывода [38], [39], [40], [41], [42]. Был предложен ряд средств защиты [43], [44], [45], [46], [47], однако многие из них оказались менее эффективными против адаптивного противника [48], [49], [50], [51]. В некоторых недавних работах предпринята попытка создать универсальные возмущения, которые вызвали бы неправильную классификацию для нескольких изображений в незараженном DNN [52], [53]. Этот направление работы рассматривает другую модель угрозы, предполагающую модель незараженной жертвы, которая не является целевым сценарием нашей защиты.

9 Заключение и будущая работа

наша работа описывает и эмпирически доказывает наш надежный и общий обнаруженный метод обнаружения против бэкдор (Trojan) атак на DNN. Помимо эффективности нашей защиты от базовых и сложных бэкдоров, одним из неожиданных выводов нашей статьи являются существенные различия между двумя методами внедрения бэкдоров: сквозной атакой с использованием BadNets управляемой триггерами, с полным доступом к обучению модели и троянской атакой, управляемой нейронами, без доступа к обучению модели. В ходе наших экспериментов мы обнаружили, что метод внедрения троянской атаки обычно добавляет больше возмущений, чем необходимо, и вносит непредсказуемые изменения в нецелевые нейроны. Это делает их триггерами сложнее перепроектировать и делает их более устойчивыми к фильтрации и нейронной обрезке. Однако компромисс заключается в том, что их сосредоточенность на конкретных нейронах делает их чрезвычайно чувствительными к смягчению последствий путем отучения. Напротив, вредность вносит более предсказуемые изменения в нейроны и может быть легче подвергнута обратному проектированию, фильтрации и смягчению с помощью обрезки нейронов.

Наконец, хотя наши результаты устойчивы к целому ряду атак в различных приложениях, все еще существуют ограничения. Прежде всего, это вопрос

обобщения за пределами текущей области видения. Наша высокоуровневая интуиция и разработка методов обнаружения/смягчения последствий могут быть обобщены: интуиция для обнаружения заключается в том, что зараженная метка более уязвима, чем незараженные метки, и это не должно зависеть от домена. Основная задача адаптации всего пайплайна к области, не связанной с видением, заключается в формулировании процесса бэкдорной атаки и разработке показателя, измеряющего, насколько уязвим сервер. конкретной меткой является (например, уравнение 2 и уравнение 3). Во-вторых, пространство потенциальных контрмер злоумышленника может быть большим. Мы изучаем 5 различных контрмер, которые конкретно нацелены на различные компоненты/предположения нашей защиты, но дальнейшее изучение других потенциальных контрмер остается частью будущей работы.