Обнаружение атаки Kerberoasting с использованием машинного обучения: от теории к практике

Оглавление

О чём статья

Привет, Хабр! Меня зовут Алексей Синадский, я руководитель исследовательских проектов R&D-центра компании UDV Group. Мы занимаемся разработкой решений в сфере информационной безопасности с применением ML. Также мы берём на борт студентов – временно на диплом и постоянно на работу. Сегодня я расскажу про результаты дипломной работы Александра Чиркина по выявлению атаки Kerberoasting в сетевом трафике, выполненной в UDV Group.

В эпоху стремительного роста угроз информационной безопасности защита корпоративных сетей становится критически важной. Одной из серьезных и довольно распространенных угроз является атака Kerberoasting, которая позволяет злоумышленнику, имеющему лишь базовые привилегии, извлечь хэш пароля сервисной учетной записи. В этой статье описывается, как методы машинного обучения помогают решению класса NTA обнаружить такую атаку в режиме реального времени. Дополнительно приводятся подробности реализации прототипа, экспериментальные результаты и ссылка на исходный код, опубликованный на GitHub.

Зачем тут ML?

Протокол Kerberos широко применяется для сетевой аутентификации, однако его особенности дают возможность проведения атак без непосредственного нарушения аутентификационных процессов. При атаке Kerberoasting злоумышленник, используя валидную учетную запись, запрашивает билеты для сервисов, из которых затем извлекает хэш пароля.

Традиционные методы обнаружения (сигнатурный анализ журналов, поиск определённых событий) часто не позволяют вовремя выявить атаку. Здесь на помощь приходят методы машинного обучения, способные анализировать статистические аномалии в трафике и вовремя обнаруживать подозрительное поведение.

Пара подходов апробированы студентом УрФУ Александром Чиркиным в рамках прохождения преддипломной практики в UDV Group, по результатам которой в соавторстве с Алексеем Синадским написана эта статья.

Суть атаки и существующие способы выявления

Атака Kerberoasting базируется на следующем механизме:

- **Запрос билета сервиса.** Пользователь обращается к центру распределения ключей (KDC) за билетом, часть которого шифруется с использованием хэша пароля сервисной учетной записи.
- **Извлечение хэша.** При использовании слабых алгоритмов (например, RC4-HMAC-MD5) злоумышленнику удаётся получить хэш, а затем провести офлайн-атаки для восстановления пароля.

Существуют классические способы обнаружения: мониторинг событий с кодом 4769 в Windows, установка honeypot’ов (специальных сервисов, не выполняющих полезных действий, но выступающих в качестве «ловушки»), сигнатуры на слабые алгоритмы шифрования. Но все эти методы имеют существенные ограничения: не всегда есть доступ к журналам логов, установка honeypot – вообще отдельная активность, сигнатуры часто пропускают атаки, либо, наоборот, формируют большое количество ложно-положительных сработок.

Некоторые средства обнаружения атак имеют способность осуществлять оконный сигнатурный анализ – поиск повторяющихся в ограниченной по времени серии пакетов определенных сигнатур. Но окно это требуется задавать экспертно на этапе создания правила, что затрудняет его адаптацию для защищаемой системы – и то при условии, что у сотрудника отдела ИБ вообще дойдут руки до кастомизации правил).

Вариант решения части проблем – построение нормального профиля сетевого трафика для Kerberos, и выявление отклонений от него.

Общая идея подхода

Разработанное решение выявляет два варианта атаки, не обнаруживаемых правилами без дополнительной адаптации к особенностям защищаемой системы, – flood-атаку и атаку с фиксированной частотой.

Обнаружение flood-атаки

При flood-атаке злоумышленник отправляет большое количество запросов за короткий промежуток времени. Для обнаружения такой активности применяется одноклассовый метод опорных векторов (One-Class SVM). Пакеты группируются по временным окнам (обычно 1 секунда), и формируется вектор признаков, содержащий:

- общее количество TGS-запросов за окно;
- среднее число запрашиваемых сервисов с одного IP-адреса.

Если модель SVM выдает аномальную метку (например, -1), система генерирует предупреждение.

Обнаружение атаки с фиксированной частотой

В этом сценарии злоумышленник отправляет запросы с равномерным интервалом. Здесь используется комбинация двух методов – LOF и статистической проверки.

С помощью алгоритма Local Outlier Factor (LOF) вычисляются показатели «выбросности» для каждого временного окна. Для подвыборки, где LOF не превышает порог (например, 1.5), применяется критерий согласия Колмогорова–Смирнова, проверяющий равномерность распределения запросов. Если равномерность подтверждается (распределение не отличается от равномерного), это сигнализирует о подозрительной активности.

Напомним суть алгоритмов

One-Class SVM

One-Class SVM «учится» на примерах нормального поведения и пытается определить границу (например, гиперсферу) вокруг этих данных. Если новая точка оказывается за пределами этой границы, она считается аномальной.

Если поверхность – сфера, то алгоритм подбирает центр c и радиус r так, чтобы большинство (доля задаётся) нормальных точек x_i удовлетворяли условию: $\|x_i - c\|^2 \leq r^2$. Если для новой точки x это неравенство не выполняется, то её относят к аномалиям.

Local Outlier Factor (LOF)

LOF оценивает локальную плотность вокруг каждой точки и сравнивает её с плотностью соседних точек. Если точка находится в области с заметно меньшей плотностью, чем её окружение, она считается аномальной.

Существует множество точек *X*. Выбирается количество *k* ближайших соседей. Для каждой точки *Xn* выбираются ближайшие соседи *Nk* и вычисляется *k-distance* – расстояние до самого дальнего из *Nk*. Затем для каждой другой точки *Xi* рассчитывается «расстояние доступности» (*reachability distance*) *RD(Xi<>n)* – как собственно расстояние между *x* и *y*, если оно больше *k-distance*, или как само *k-distance*, если меньше. Затем для исходной точки *Xn* рассчитывается «плотность доступности» (*local reachability density*) *LRD(Xn)* как инвертированное среднее расстояние до k-ближайших соседей (*RD* выше). И после этого вычисляется уже *LOF* для каждого из *X* как средняя плотность доступности для каждого из k-соседей для этой точки (среднее *LRD(Nk)*), умноженная на инвертированную плотность доступности для исследуемой *Xn*.

$$RD(X_n, X_i) = \max(k_{distance}(X_i), distance(X_n, X_i))$$

$$LRD(X_n) = \frac{k}{\sum_{X_j \in Nk} RD(X_n, X_i)}$$

$$LOF(X_n) = \frac{\sum_{X_i \in Nk} LRD(X_i)}{k \times LRD(X_n)}$$

Упрощая, можно сказать, что это отношение средней плотности соседей к плотности исследуемой точки.

Если LOF существенно больше 1, точка отличается от окружающих и считается выбросом.

Подробности реализации

Исходный код разработанного прототипа опубликован на GitHub: https://github.com/UDV-RnD/Articles/tree/main/A.Sinadskiy. Detecting a Kerberoasting Attack using Machine Learning - from theory to practice/kerberoasting_ml_detector.

Рассмотрим основные компоненты.

Сбор и предварительная обработка данных

Модуль, реализованный на базе библиотеки scapy, обеспечивает перехват сетевых пакетов с KDC (на 88 порт). Возможна адаптация к чтению Zeek-логов.

```
from scapy.all import sniff, UDP

def packet_callback(packet):
    if packet.haslayer(UDP) and packet[UDP].dport == 88:
        process_packet(packet)

sniff(filter="udp port 88", prn=packet_callback, store=0)
```

Из каждого пакета извлекаются ключевые поля (время запроса, IP-адрес источника, идентификатор сервиса (поле sname). Полученные данные группируются в окна фиксированной длительности (например, по 1 секунде), что позволяет формировать обучающие векторы.

```
import numpy as np
from collections import defaultdict

def group_packets_by_window(packets, window_size=1):
    windows = defaultdict(list)
    for pkt in packets:
        t = int(pkt.time)
        windows[t // window_size].append(pkt)
    return windows
```

Для каждого временного окна вычисляются статистические характеристики:

- общее количество TGS-запросов;
- среднее число запрашиваемых сервисов на одного клиента.

Эти признаки формируют векторы для обучения моделей:

```
def extract_features(window_packets):
    total_requests = len(window_packets)
    services_per_ip = {}
    for pkt in window_packets:
        ip = pkt[IP].src
        service = pkt.sname if hasattr(pkt, "sname") else None
        if service:
            services_per_ip.setdefault(ip, []).append(service)
    avg_services = np.mean([len(svcs) for svcs in services_per_ip.values()]) if
services_per_ip else 0
    return [total_requests, avg_services]
```

Модели машинного обучения

Для обнаружения flood-атаки применяется One-Class SVM из библиотеки scikit-learn:

```
from sklearn.svm import OneClassSVM

svm_model = OneClassSVM(kernel='rbf', nu=0.1)
svm_model.fit(features_train)
predictions = svm_model.predict(features_test)
```

Для анализа атаки с фиксированной частотой используется алгоритм LOF, а затем проверяется равномерность с помощью критерия Колмогорова–Смирнова из модуля scipy.stats:

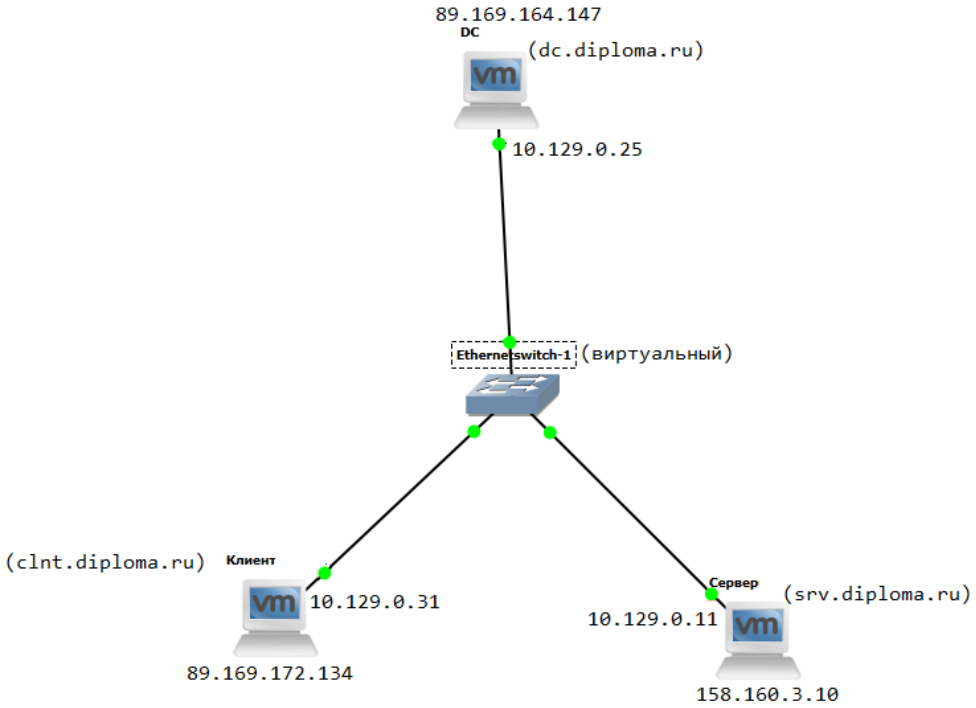
```
from sklearn.neighbors import LocalOutlierFactor
from scipy.stats import kstest

lof = LocalOutlierFactor(n_neighbors=50)
lof_scores = lof.fit_predict(features)
```

```
# Здесь window_intervals – список интервалов между запросами
statistic, p_value = kstest(window_intervals, 'uniform')
if p_value > 0.05:
    # Интервалы распределены равномерно – возможно, атака с фиксированной частотой
    flag_attack()
```

Экспериментальные результаты

Тестовый стенд и датасет

Для оценки работоспособности разработанного подхода подготовлен тестовый стенд, состоящий из трех виртуальных машин: KDC, Клиент, Сервер приложений (с развернутым SSH-сервером, поддерживающим аутентификацию по протоколу Kerberos).



На этом стенде был сгенерирован тестовый датасет, описывающий трафик около 1000 пользователей и содержащий нормальный трафик, flood-атаку, атаку с фиксированным интервалом. Ссылка на датасет: https://github.com/UDV-RnD/Articles/tree/main/A.Sinadskiy. Detecting a Kerberoasting Attack using Machine Learning - from theory to practice/kerberoasting_ml_detector/datasets

Нормальный трафик содержит запросы пользователей, выполняемые в рамках обычной работы, когда запросы на получение билетов (AS_REQ, TGS_REQ) соответствуют привычной активности.

При flood-атаке с одного клиента за короткий промежуток времени отправляется аномально большое количество TGS_REQ запросов. Для таких записей в датасете устанавливается метка *flood_mark = 1*.

Для записи датасета с атакой с фиксированным интервалом злоумышленник отправляет запросы с равномерно, с повторяющимся интервалом (например, каждую секунду или с другой постоянной периодичностью). Такие события размечаются меткой *fixed_time_mark = 1*.

Каждая запись в датасете включает информацию о времени запроса, IP-адресе клиента, идентификаторах клиентов и сервисов, а также другие служебные параметры. После генерации датасета данные могут быть дополнительно размечены и интегрированы с историческими данными для дальнейшего обучения модели.

Результаты выявления

Обе предложенные модели показали низкое количество ложных срабатываний (False Positive Rate, FPR). При этом обе они выявляют не все события истинных атак, но имеющиеся сработки позволяют «поднять флаг» о том, что в сети что-то не так, и указать на подозреваемый тип зловредной активности.

Минимизировать FPR для атаки с фиксированной частотой удалось за счёт фильтрации предварительных выбросов (порог LOF – 1.5) и статистической проверки распределения (критерий Колмогорова–Смирнова).

Численно:

Атака\Метрика	Precision	Recall	FPR
Flood-атака	0.5	1.0	0.0002%
Фикс. частота	1.0	0.3	0%

Заключение

Применение предложенных подходов, основанных на методах машинного обучения, позволяет разрабатывать методы обнаружения, обеспечивающие возможность выявления аномальной Kerberos-активности, которая может быть атакой. От существующих правил обнаружения предложенные алгоритмы отличает адаптивность, благодаря которой особенности работы системы могут быть учтены в детекторе без непосредственного участия человека-оператора.

Реализованное решение, описанное в дипломной работе и подтверждённое экспериментами, демонстрирует достаточную точность обнаружения как для flood-атак, так и для атак с фиксированной частотой.

Исходный код, полностью воспроизводящий описанную методику, доступен на GitHub: https://github.com/UDV-RnD/Articles/tree/main/A.Sinadskiy. Detecting a Kerberoasting Attack using Machine Learning - from theory to practice/kerberoasting_ml_detector.

Данный подход может применяться в решениях класса NTA. Разумеется, для этого полученный PoC необходимо доработать. Дальнейшие планы включают добавление ещё больше адаптивности с помощью автоматического подбора порогов, оптимизацию модели для снижения количества ложных срабатываний и увеличение количества признаков для выявления дополнительных вариантов реализации атаки.

Внедрение системы на базе предложенного PoC и сгенерированного датасета даёт практическую возможность раннего обнаружения атак Kerberoasting, что помогает снизить риск компрометации сервисных учетных записей и последующей эскалации привилегий в сети.