1 背景介绍

数据访问策略的错误配置已经成为导致安全事件发生的主要原因。当用户抱怨自己被拒绝访问时，系统管理员需要立即解决用户的问题，就可能存在过多授权的问题。

Time	Incident	Organization
2017.6	198 million US voter records leaked [39]	Deep Root Analytics
2017.7	14 million customer records leaked [42]	Verizon
2017.9	Half million vehicle records leaked [28]	SVR Tracking
2018.2	119,000+ personal IDs exposed [29]	FedEx
2018.3	42,000 patients information leaked [17]	Huntington hospital
2018.4	63,551 patients records breached [16]	Middletown medical
2019.1	24 million financial records leaked [19]	Ascension
2019.9	20 million citizen records exposed [76]	Novaestrat

Table 1: Recent publicly-reported security incidents caused by access control misconfigurations.

图 1: 框架

2 基于现有访问控制机制观察

本文创新：

1 不同的软件系统实现了不同的访问控制模型，即使是相同的访问控制模型，策略配置的语法和格式也可能存在不同。

2 不同的访问控制日志都有统一的格式而且比较容易解析。都可以表示为四元组¡S,O,A,R¿，其中S表示subject，O表示object，A表示action，R表示result（result分为allow和deny）。所有的访问控制策略都可以使用if-then语句表示。基于以上几点，作者想到了用决策树来处理。但是传统的决策树无法解决带时间序列的问题、策略更新的编码问题。

3 P-DIFF模型

主要解决的三个问题：

1 怎么去维系策略策略改变历史？-Time-Changing Decision Tree (TCDT)

2 如何从访问日志中推断访问策略？-a decision-tree-based learning algorithm

3 如何管理改变之后的策略？

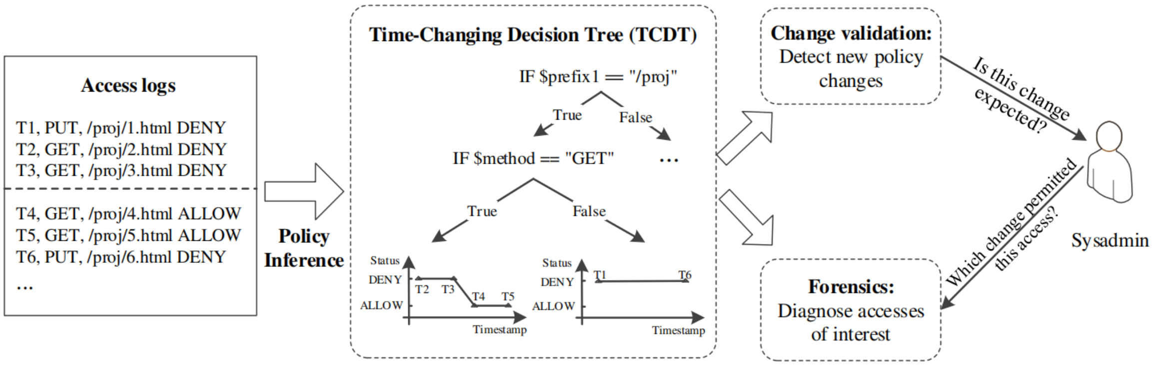

图 2: *P-DIFF*模型

4 DT和TCDT

1 decision tree：节点分为两种：内部节点和叶子节点。内部节点表示（属性名属性值）。subject、object和action都可以是属性。结果是(r,pr),r表示结果，pr表示结果的概率。

2 time-changing decision tree：结果表示为：

$$T = ((\tau_1, r_1), (\tau_n, r_n), ..., (\tau_n, r_n)) \tag{1}$$

引入了时间序列的概念。

5 一种基于决策树的学习算法

特征抽取： 算法步骤：

Field	Annotation	Semantics
Timestamp	%t	Timestamp of each access
Hierarchical feature	%h(*)	Features with hierarchical namespace, such as IP address, URL, etc. * is a delimiter character.
Normal feature	%n	Non-hierarchical features
Access result	%l	ALLOW or DENY
Irrelevant	%o	Irrelevant fields

Table 3: Annotations of the log format. P-DIFF requires users to annotate the access log format, which is a one-time effort for a given system.

图 3: 特征提取

Algorithm 1 **Decision Tree Learning**

```
function DTL(L) ^a
    root ← treenode()
    i, x_{i_j} ← best_split(L) ^b
    L_l, L_r ← split(L, i, x_{i_j}) ^c
    mg ← metric_gain(L, L_l, L_r) ^d
    if mg! = 0 then
        root.left ← DTL(L_l)
        root.right ← DTL(L_r)
    return root
```

[a] $L = \{(x_{i_1}, \ldots, x_{i_n}, y)|i \in [1, m]\}$, the training data.
[b] Find the feature j and its value x_{i_j} that split L into two purest subsets, i.e. subsets with as large proportion of ALLOW or DENY as possible.
[c] Split L into $L_l = \{(x_{k_1}, \ldots, x_{k_n}, y)|k \in [i, m] \wedge x_{k_j} = x_{i_j}\}$ and $L_r = L - L_l$.
[d] Calculate $metric(L_l)$+$metric(L_r)$-$metric(L)$, where metric is a function measures the label purity of a set, e.g. entropy or Gini Impurity.

图 4: 算法步骤

6 策略改变管理

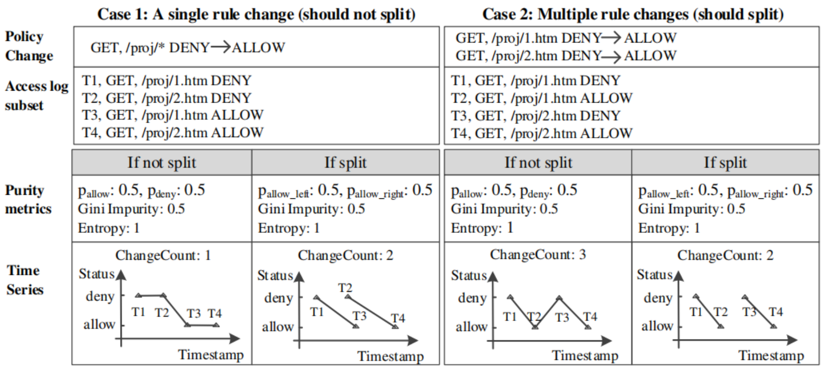

Figure 7: Examples that demonstrate splitting events in TCDT-based policy learning (cf. §8). Case 1 does not require splitting, while Case 2 does due to the condition: `if prefix2=="/proj/1.htm"`. Traditional splitting metrics cannot decide whether to split if a change is involved, because the possibility of ALLOW or DENY is always 0.5 in each subset (Gini Impurity: $1-(p_{allow})^2-(p_{deny})^2=0.5$, Entropy: $-p_{allow}\log(p_{allow})-p_{deny}\log(p_{deny})=1$). The time-series change counts differ in the subsets and can guide correct splitting events.

图 5: 算法步骤

7 优化

优化点：1）循环中只计算deny的值 2）离散卷积的方法

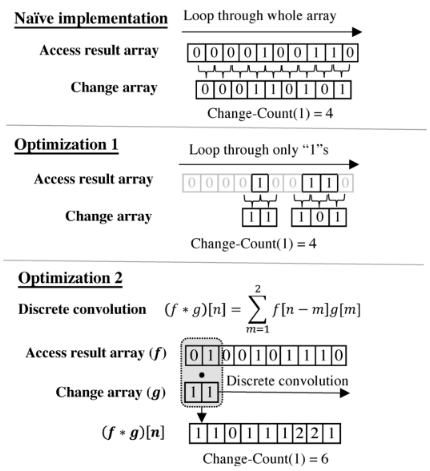

图 6: 算法步骤

8 实验评估

TCDT：准确率：0.997 召回率：0.92 F-score：0.94

spark-dl:准确率：0.83 召回率：0.86 F-score：0.80

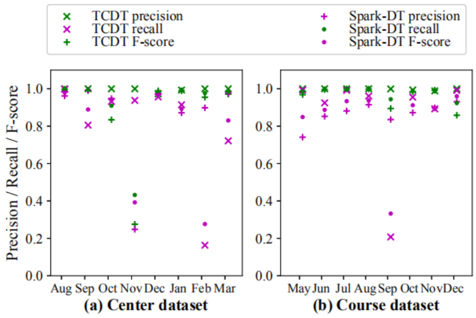

Figure 11: Precision, recall, and F-score of TCDT classifying access results for the Center and Course datasets. The x-axis shows the time of the testing data, which is a month of logs in the dataset. The training data is the three continuous months of logs before the testing month.

图 7: 算法步骤