

# 10 基于主机的入侵检测技术

2019年5月30日 16:03

## [1 审计数据的获取](#)

- [1.1 审计数据类型和来源](#)
- [1.2 审计数据的预处理](#)
- [1.3 审计数据获取模块的设计](#)

## [2 用于入侵检测的统计模型](#)

## [3 入侵检测的专家系统](#)

## [4 基于状态转移分析的入侵检测技术](#)

## [5 文件完整性检查](#)

## [6 系统配置分析技术](#)

## [1 审计数据的获取](#)

- 审计数据的获取是主机入侵检测技术的重要基石，是进行主机入侵检测的信息来源。审计数据的获取质量和数量，决定了主机入侵检测工作的有效程度。审计数据获取需要考虑的问题：
  - 确定审计数据的来源和类型
  - 审计数据的预处理工作，其中包括记录标准格式的设计、过滤和映射操作等
  - 审计数据的获取方式，包括审计数据获取模块的结构设计和传输协议等
- [1.1 审计数据类型和来源](#)
- [1.2 审计数据的预处理](#)
- [1.3 审计数据获取模块的设计](#)

### 1.1 审计数据类型和来源

- 根据目标系统的不同类型和主机入侵检测的不同要求，所收集审计数据的类型也不尽相同：
  - 从目标主机的类型来看，不同操作系统的审计机制设计存在差异，主机活动的审计范围和类型也不同
  - 根据不同主机入侵检测系统的设计要求和需求，其具体选取的审计数据的类型和来源也各有侧重
- 以IDES系统为例，IDES在SunUNIX目标系统环境下所收集到的审计数据主要分为以下四类：
  - 文件访问：包括对文件和目录进行的操作，如读取、写入、创建、删除和访问控制列表的修改
  - 系统访问：包括登录、退出、调用以及终止超级用户权限等
  - 资源消耗：包括CPU、I/O和内存的使用情况
  - 进程创建命令的调用：指示一个进程的创建

### 1.2 审计数据的预处理

- 主机入侵检测所要进行的主要工作就是审计数据的预处理工作，包括映射、过滤和格式转化等操作。预处理工作体现在以下几个方面：
  - 不同目标系统环境的审计记录格式各不相同，对其进行格式转化的预处理操作形成标准记录格式后，将有利于系统在不同目标平台系统之间的移植；同时，有利于形成单一格式的标准审计记录流，便于后继的处理模块进行检测工作
  - 对于审计系统而言，系统中发生的所有可审计活动都会生成对应的审计记录，因此对某个时间段而言，审计记录生成速度非常快，而其中往往大量充斥着对于入侵检测而言无用的事件记录，所以需要对审计记录进行必要的映射和过滤等操作

### 审计数据的预处理—标准审计记录格式设计

- 以IDES系统为例，IDES审计记录格式是基于若干考虑设计的：
  - 它必须通用程度足够高，以便能够表示目标监控系统的所有可能事件类型
  - 它应该是机器中最有效的数据表示格式，以将处理开销降低到最小程度
  - 记录格式应该按标准化设计，使IDES能够从多个不同类型的机器处接收输入记录，而无须进行任何的数据转换

- IDES的功能取决于其所接收到的信息，因此每一个审计记录中应该尽可能地提供更多的信息。在IDES审计记录中所有能够被填入相关信息的字段都应该被填写
- 并不是所有的可检测事件都需要报告给IDES，例如SunOS系统中就不将STAT(2)系统调用报告给IDES，其原因为：
  - 这些系统调用很频繁，并通常是冗余的
  - STAT(2)调用的量很大，忽略它将能够显著提高系统运行性能
- STAT系统的标准审计记录格式由三部分组成：
  - <Subject,Action,Object>每一部分都进一步包括更详细的字段定义

subject	Us_ruid	真实用户ID
	Us_euid	有效用户ID
	Us_gid	用户组ID
action	Us_action	用户操作类型
	Us_time[2]	操作时间戳
	Us_pid	进程ID号
object	Us_objname	对象文件
	Us_perm	访问权限
	Us_owner	属主
	Us_gowner	用户组属主
	Us_inode	i-node号
	Us_dev	设备号
	Us_fsid	文件系统ID号
	Us_target	目标文件

- BSM审计记录到STAT审计记录的映射关系如表所示

Process audit ID	→	Us_ruid
Process user ID	→	Us_euid
Process real group ID	→	Us_gid
Header event type	→	Us_action
Path path argument	→	Us_objname
Attribute mode	→	Us_perm
Attribute uid	→	Us_owner
Attribute gid	→	Us_gowner
Attribute nodeid	→	Us_inode
Attribute rdev	→	Us_dev
Attribute fside	→	Us_fsid
Path path argument	→	Us_target
Header time of queue	→	Us_time[2]
Process process ID	→	Us_pid

### 1.3 审计数据获取模块的设计

- 审计数据获取模块的主要作用是获取目标系统的审计数据，并经过预处理工作后，最终目标是为入侵检测的处理模块提供一条单一的审计记录块数据流，供其使用
- 设计时需要考虑的问题：
  - 在各目标主机系统和中央分析控制台之间处理负荷的平衡问题
  - 采用何种传输协议
  - 如何将从多个目标主机获得的审计数据多路复用成为一条单一审计记录数据流的问题
  - 如何处理诸如网络传输过程中可能出现的延时、遗漏等问题

## 2 用于入侵检测的统计模型

- Denning在1986年经典论文中提出了四种可用于入侵检测的统计模型
  - 操作模型

- 该模型主要关心对系统中所发生事件的计数度量情况，例如观察在特定时间间隔内发生的失败登录事件的次数等
  - 通常的模型操作包括将所关心的特定事件计数值与某个阈值进行比较，如果超过，则指示发生了异常情况
  - 该模型可以同时应用到异常入侵检测和滥用入侵检测技术中
- 均值和标准偏差模型
- 该模型成立的一个假设基础：
    - 系统当前状态特征可以采用数据的均值和标准偏差两个度量参数来刻画
    - 在检测过程中，如果当前用户行为超出了可信任的区间范围，则表示为异常行为
    - 信任区间的定义通常采用参数度量与其平均值的标准偏差值
- 多元模型
- 该模型是对均值和标准偏差模型的扩展，其主要思想是：
    - 在多个参数度量之间进行相关分析，从而摆脱单纯依赖单个度量值来判断系统当前状态的限制因素
- 马尔可夫过程模型
- 四个统计模型中最复杂，其基础思想是：
    - 将每个审计记录中不同类型事件的出现视为随机变量的不同取值
    - 采用随机过程模型来刻画入侵检测系统的输入事件流
    - 模型中采用状态转移矩阵来表示不同事件序列出现的概率值，如果当前审计事件按照正常的状态转移矩阵所计算出的发生概率小于某个阈值，则指示为异常行为
- 四个模型比较：
- 第一个模型“操作模型”的典型应用包括入侵检测中的阈值检测或者启发式阈值检测技术等
  - 第二和第三个模型在经典的IDES系统中得到很好的实现
  - 第四个模型在TIM系统中得到较好的体现
- 以IDES/NIDES为例介绍其系统中使用的统计分析技术
- IDES统计异常检测引擎观测在所监控计算机系统上的活动行为，并自适应地学习主体的正常行为模式。所定义的主体模型包括：
- 单个用户
  - 用户组
  - 远程主机
  - 整个系统
- 所观测的行为如果与所期望该主体的行为偏离显著时，就被标识为潜在的入侵行为
- 在IDES系统的统计分析组件中，当前系统的活动状态采用一组测量值参数变量来表示，称为“入侵检测向量”
- 根据目标主体类型的不同，IDES定义了3种不同类型的测量值，分别针对用户主体、目标系统主体和远程主机主体
- 根据目标主体类型的不同，把IDES统计分析系统中不同类型的单独测量值分为以下4个类别：
- 活动强度测量值
  - 审计记录分布测量值
  - 类别测量值
  - 序数测量值
- 不同类型的测量值用于不同的测量目的：
- 活动强度测量值
    - 保证所生成的活动流量正常
  - 审计记录分布测量值
    - 保证在最近生成的数百个审计记录中，发生的活动类型正常
  - 类别测量值和序数测量值
    - 确保在一个活动类型中在最近生成的数百个审计记录中产生该动作的行为正常
- 以IDES系统中应用于用户主体类型的测量值为例，它们包括下列定义的类型：
- 序数测量值：CPU使用情况、I/O使用情况
  - 类别测量值：使用物理位置、邮件程序使用、编辑器使用、编译器使用、外壳使用、窗口命令使用、通用程序使用、通用系统调用使用、目标活动、目录使用、文件活动、文件使用、所访问用户的ID号、系统错误类型、以小时计的审

计记录活动、以天计的使用情况、远程网络活动类型、远程主机网络活动、本地网络活动类型、本地主机网络活动等

- 审计记录分布测量值:
  - 对于以上的每一个测量值，在审计记录分布测量值中都有一个对应的类别
- 活动强度测量值:
  - 每分钟流量
  - 每10分钟流量
  - 每小时流量

### 3 入侵检测的专家系统

- 专家系统并非为入侵检测而设计，但是在最早期的若干入侵检测系统中就已经开始使用专家系统。专家系统在知识库的基础上，根据所获得的事实和已知的规则进行推导，并得出结论
- 专家系统只能基于明确的、可靠的规则得出结论，对于超出已有规则范围的事实，它无法得出有用的结论
- 在构建专家系统之前，用户必须首先定义规则（Rule）和模式类型（ptype）
  - Ptype[count value:int]
  - 以上声明语句包括一个关键词ptype、方括号、一个名称count和一个类型字段value，该字段声明为一个整数类型。整个声明以方括号结尾
  - 以上声明的目的在于建立一个关于事实的模式（Pattern）或模板（Template）
- 对于要加入到知识库的事实，必须进行“断言（assert）”操作
- 构建一个输入接口的必要步骤：
  - 为用户所需要加入到知识库中的事实作出对应的ptype类型声明
  - 编写一个C语言函数，来调用正确的assert\_<ptypename> () 函数
  - 编写一条规则，在其前提或者结论语句中加入对此C语言函数的调用

### 4 基于状态转移分析的入侵检测技术

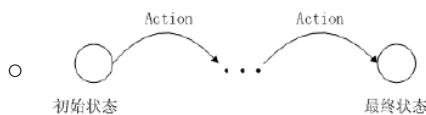
- 最初的基于状态分析的入侵检测模型是在20世纪90年代初由美国加州大学圣巴巴拉分校的提出并实现
- 原型系统最初在UNIX环境下实现，称为USTAT系统，进一步扩展到网络环境下的多主机检测情况，称为NSTAT系统
- 最新一代的采用STAT技术的系统称为NetSTAT系统
- 特点：
  - 脱离了单纯进行主机入侵检测的结构，实现了分布式的入侵检测框架
- 工作原理：
  - 基于状态分析的检测模型，将攻击者的入侵行为描绘为一系列的特征操作及其引起的一系列系统状态转换过程，从而使得目标系统从初始状态转移到攻击者所期望的危害状态
- 优点
  - 直接采用审计记录序列来表示攻击行为的方法不具备直观性，它需要专门的知识才能理解一个具体的攻击行为所对应的审计记录序列，从而使得快速直观的更新检测规则库非常困难，而STAT采用高层的状态转移表示法来表示攻击过程，避免了这一问题
  - 对于同一种攻击行为可能对应着不同的审计记录序列，这些不同的审计记录序列可能仅仅因为一些细微的差别而无法被采用直接匹配技术的检测系统察觉，而STAT系统可以较好地处理这种情况
  - STAT能够检测到由多个攻击者所共同发起的协同攻击，以及跨越多个进程的攻击行为
  - STAT的一大特色就是具备在某种攻击行为尚未造成实际危害时，就及时的检测到并采取某种响应措施的能力
- 从本质上讲STAT系统属于基于规则分析的滥用入侵检测系统，不仅能检测到已知的攻击类型，还能够较好的处理已知攻击类型的变种行为

#### 状态转移图

- 由两个基本组件构成：
  - 结点：表示系统状态
  - 弧线：表示特征行为
- 状态转移图的基本组件如下图所示：



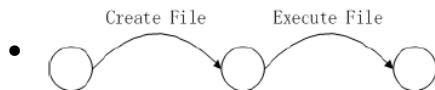
- 状态转移图通常包括一个初始状态、一个最终状态以及若干攻击行为步骤，及其引起的若干中间状态，如下图所示：



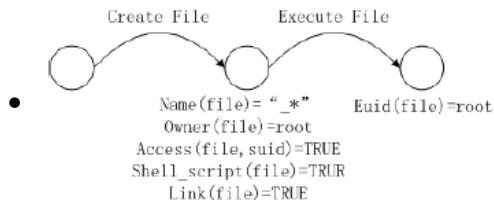
- 以一个简单例子来展示状态转移图的构成过程。该例子代表着UNIX环境下的一次攻击行为，其步骤如下：

```
% ln target -x
%
% -x
```

- 文件target代表一个setuid的shell脚本，其属主为根用户（root），文件target中首行包含字符串“#!/bin/sh”
- 上述攻击过程说明如下：
  - 攻击者对属主为root且具有setuid特性的shell脚本target，创建一个硬连接(hard-link)文件，该连接文件的名称以字符‘-’开头
  - 攻击者执行该连接文件“-x”
- 攻击过程分析：
  - 每当创建一个硬连接文件时，系统都会创建一个新的目录项，且该目录项具备与原文件系统的访问权限和属主信息。连接文件和原目标文件共享相同的文件i-node号，因此对此连接文件的访问等同于对原目标文件的访问
  - 对内含#!/bin/sh字符串的shell脚本程序的调用，将会生成一个子shell程序
  - 如果脚本程序名称前缀为“-”，则生成的子shell将会进入交互式操作的状态，即调用脚本程序的用户获得了一个可交互式输入命令的shell程序
  - 在上述的攻击过程中，因为调用的是属主为root的setuid文件，所以用户最终获得了具备root访问权限的shell
- 创建状态转移图的过程
  - 用户创建一个所需的文件，之后执行这个文件。两个关键行为可以表示为“CREATE”和“EXECUTE”。



- 确定这些关键操作所引起的状态变化

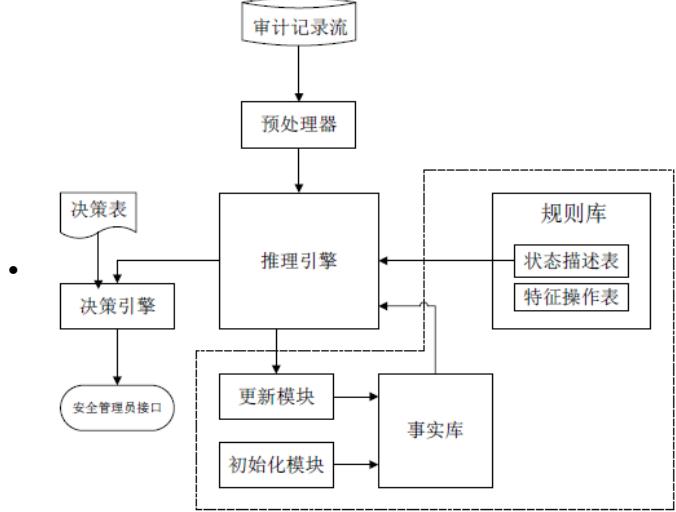


- 构建状态转移图的步骤：
  - 分析具体的攻击行为，理解内在机理
  - 确定攻击过程中的关键行为点
  - 确定初始状态和最终状态
  - 从最终状态出发，逐步确定所需的各个中间状态及其满足的状态断言组

从系统设计角度看，STAT系统包含如下的构成组件：

- 预处理器
- 知识库
  - 事实库
    - 初始化模块
    - 更新模块
  - 规则库
    - 状态描述表
    - 特征操作表
- 推理引擎
- 决策引擎

STAT系统结构示意图：



### STAT检测引擎工作原理:

- 推理引擎是整个系统的核心部件，如下所示：

	S1	S2	S3	S4	S5
1					
2					
3					
4					
...	...	...	...	...	...
n					

推理引擎表

- 如上图所示，每一行都对应着某个具体攻击过程的活动实例，而每一列则代表着攻击过程的某个具体步骤，指示着本次攻击实例的完成程度
- 可以想见，推理引擎的初始行数应该等于规则库中所表示状态转移图的数目，即推理引擎表的每一行对应着某个状态转移图的表现实例。当推理引擎工作时，每次接收到一个新审计记录后，将首先判断当前哪个状态转移图的实例匹配当前的特征操作和转移状态，在表中找到该行后，将复制一个新行并加入到现有的推理引擎表中，并在对应的表格单元处进行标注；对匹配进行复制而不是对原行进行操作的原因是原来的表项仍然可以代表某次部分完成的具体攻击实例

### 举例说明：



假想的状态转移图

	S1	S2	S3	S4	S5
1					
2					
...	...	...	...	...	...
h					
...	...	...	...	...	...
n					

初始的推理引擎表

	S1	S2	S3	S4	S5
1					
2					
...	...	...	...	...	...
h					
...	...	...	...	...	...
n					
n+1	X				

第一个操作后的推理引擎表

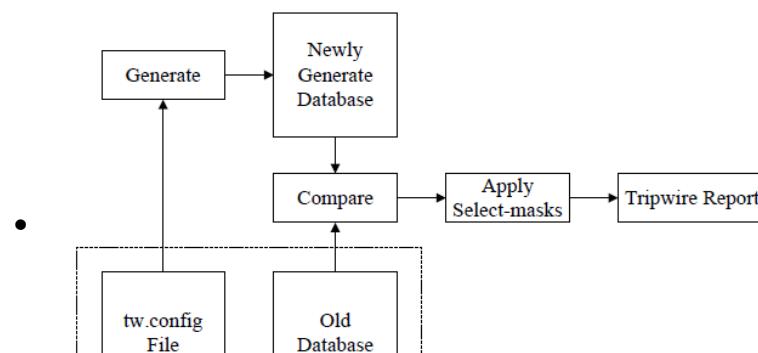
	S1	S2	S3	S4	S5
1					
2					
...	...	...	...	...	...
h					
...	...	...	...	...	...
n					
n+1	X				
n+2	X	X			

第二个操作后的推理引擎表

- 推理引擎的最重要的功能就是判断状态断言的真假，即判断当前系统状态是否满足某个特定状态转移图中某个时刻的状态值
- 为了判断是否满足某个状态的条件，推理引擎需要两个来源的输入：
  - 构成某个状态条件的断言组
  - 指示当前操作状态的审计记录

## 5 文件完整性检查

- 目的：检查主机系统中文件系统的完整性，及时发现潜在的针对文件系统的无意或恶意的更改
- 检查文件系统完整性的必要性包括以下几个方面：
  - 攻击者在入侵成功后，经常在文件系统中安装后门或木马程序，以方便后继的攻击活动
  - 攻击者还可能安装非授权的特定程序，并且替换掉特定的系统程序，以掩盖非授权程序的存在
  - 为了防止攻击活动痕迹的暴露，攻击者还可能删除若干重要系统日志文件中的审计记录
  - 入侵者还可能为了达成拒绝服务攻击目的或破坏目的，恶意修改若干重要服务程序的配置文件或者数据库数据，包括系统安全策略的配置信息等
- 基本思想：
  - 对要检查的每个目标文件生成一个唯一标识符，并将它们存储到数据库中
  - 进行检查时，对每个目标文件重新生成新的标识符，并将新标识符与数据库中存储的旧版本进行比较
  - 确定文件是否发生了更改
  - 对数据库中条目数目的检查，也可能发现文件系统中文件数目的增删变化
- 检查列表技术：
  - 特点：
    - 使用一个检查列表来存储目标文件的标识信息，列表中的每个条目包括目标文件的诸多属性信息，例如文件长度、最后修改时间、属主信息等
  - 方法：
    - 检查时，系统再重新取得目标文件的属性信息，并与在检查列表中的对应项目进行比较
  - 优点：
    - 检查速度比较快
  - 缺点：
    - 无法确保文件内容不被恶意更改，因为入侵者可在获得较高权限的条件下，更改文件内容而不改变文件的属性信息
- 完整性检查领域的最著名工具——TripWire：
  - 基本思想：
    - 使用单向消息摘要算法，计算每个文件的校验和特征信息
    - 将其存储到可靠的安全存储介质上
    - 系统定时地计算目标文件的校验和特征，并与预先存储的特征信息进行比较，如果出现了差异，则向系统管理员发出报告信息
  - 系统设计模块示意图：



- 完整性检查归为主机入侵检测技术的范围，原因如下：
  - 入侵者攻击的诸多后果往往体现在文件系统中，文件完整性检查技术实质上属于一种事后的入侵检测技术，针对的是特定文件对象的完整性问题

## 6 系统配置分析技术

- 系统配置分析（静态分析）的技术目标是检查系统是否已经受到入侵活动的侵害，或者存在有可能被入侵的危险
- 静态分析技术通过检查系统当前的配置情况，来判断系统的当前安全状况
- 之所以称为“静态”，是因为该技术只检查系统的静态特征，并不分析系统的活动情况
- 配置分析技术的基本原理基于如下两个观点：
  - 一次成功的入侵活动可能会在系统中留下痕迹，这可以通过检查系统当前的状态来发现
  - 系统管理员和用户经常会错误的配置系统，从而给攻击者入侵的可乘之机
  - 可以看出，配置分析技术既可以在入侵行为发生之前使用，作为一种防范性的安全措施；也可以使用在潜在的攻击活动之后，以发现暗藏的入侵痕迹，从这个意义上，可以将它归入主机入侵检测技术的范围
- 系统配置分析技术著名实现工具—COPS系统，其所检查的系统安全范围包括如下类型：
  - 检查文件、目录和设备的访问权限模式
  - 脆弱的口令设置
  - 检查口令文件和组用户文件的安全性、格式、内容
  - 检查在/etc/rc\*目录和cron中指定运行的文件和程序
  - 具有rootSUID属性的文件，检查它们是否可写，以及是否脚本程序
  - 对重要的二进制文件和其他文件计算CRC校验和，检查是否发生更改
  - 检查用户主目录下文件是否可写
  - 是否具有匿名FTP登录服务账户
  - 是否存在TFTP服务，Sendmail中别名情况以及在inetdconf文件中隐藏的启动脚本程序等
  - 各种类型的根权限检查
  - 按照CERT安全报告的发布日期，检查关键文件是否已经及时进行了升级或打上补丁