网络安全

第九章 入侵检测

黄 玮

温故

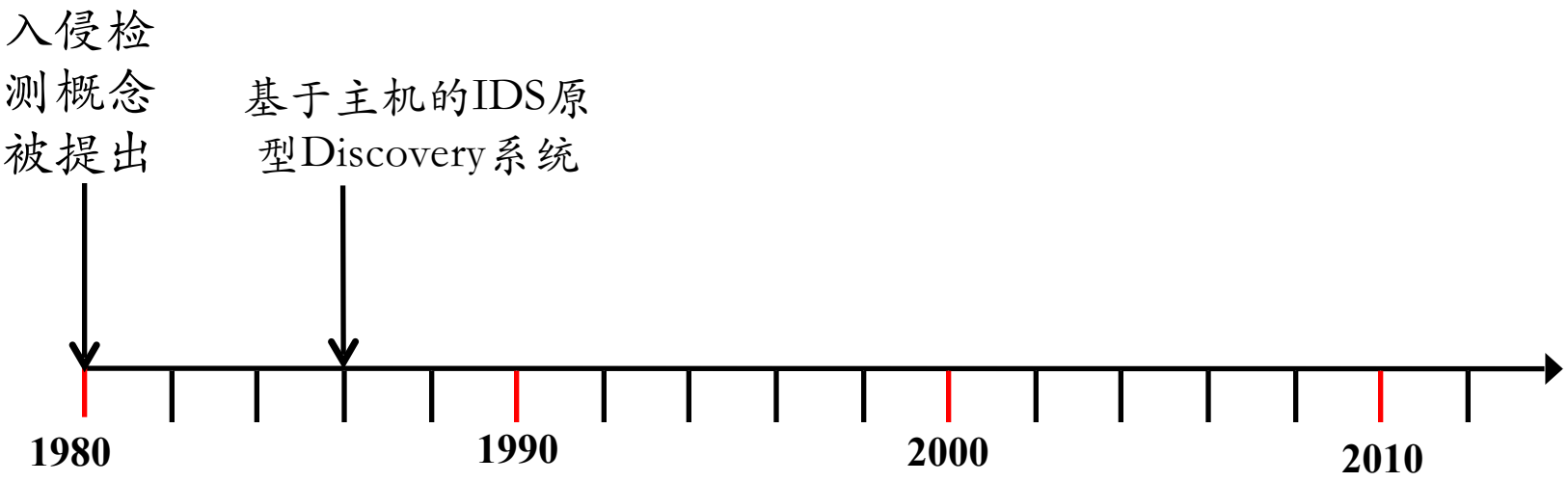
- 防火墙在网络与系统防御中的作用和地位
  - —在两个信任程度不同的网络之间设置的、用于加强访问控制的软硬件保护措施
  - —防外不防内
- 防火墙的实现技术
  - —软件技术
  - —硬件架构
    - – NP / ASIC / x86
- iptables的配置

- 应用防火墙的实现原理
  —入侵检测的核心技术原理
- 入侵检测系统和防火墙的关系
- 入侵检测系统的应用场景

本章内容提要

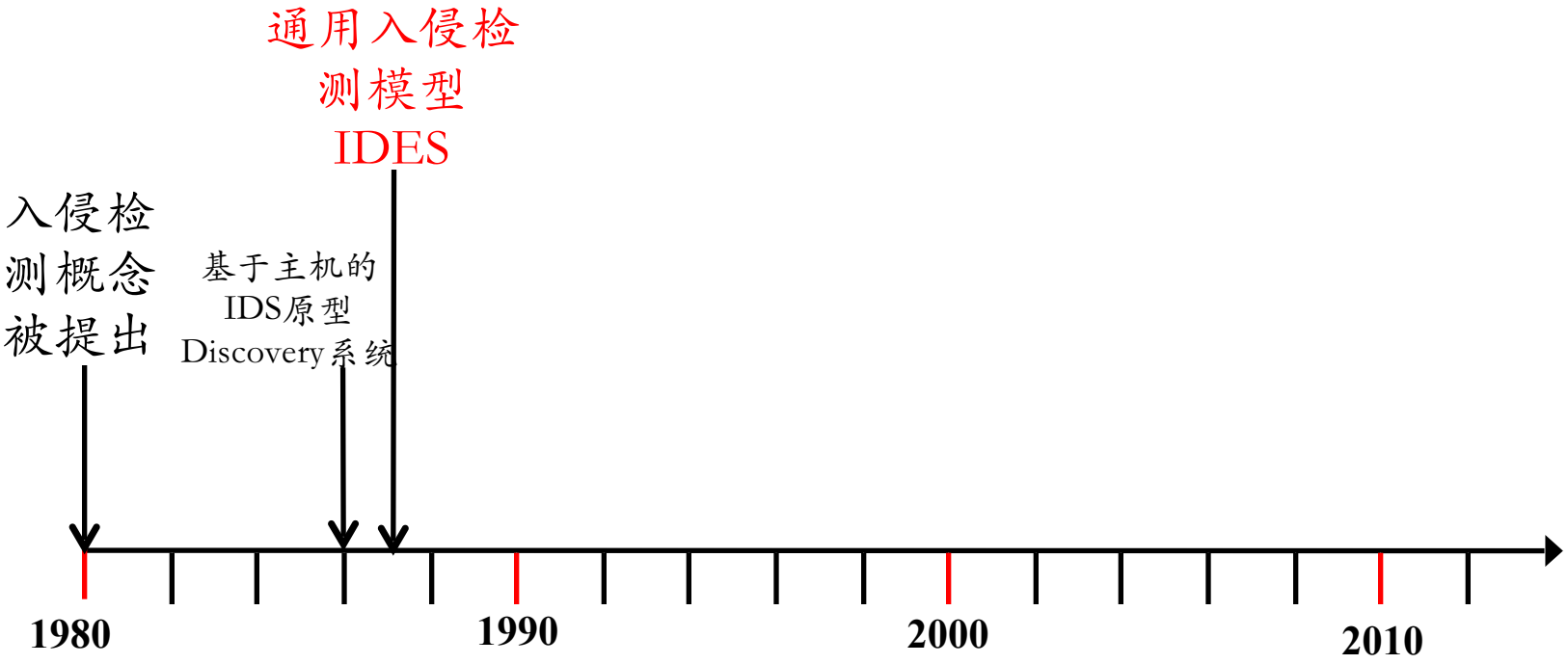
- 入侵检测发展史
- 入侵检测理论
- 入侵检测关键技术
- 入侵检测标准化
- 入侵检测系统配置

入侵检测发展简史（1/5）



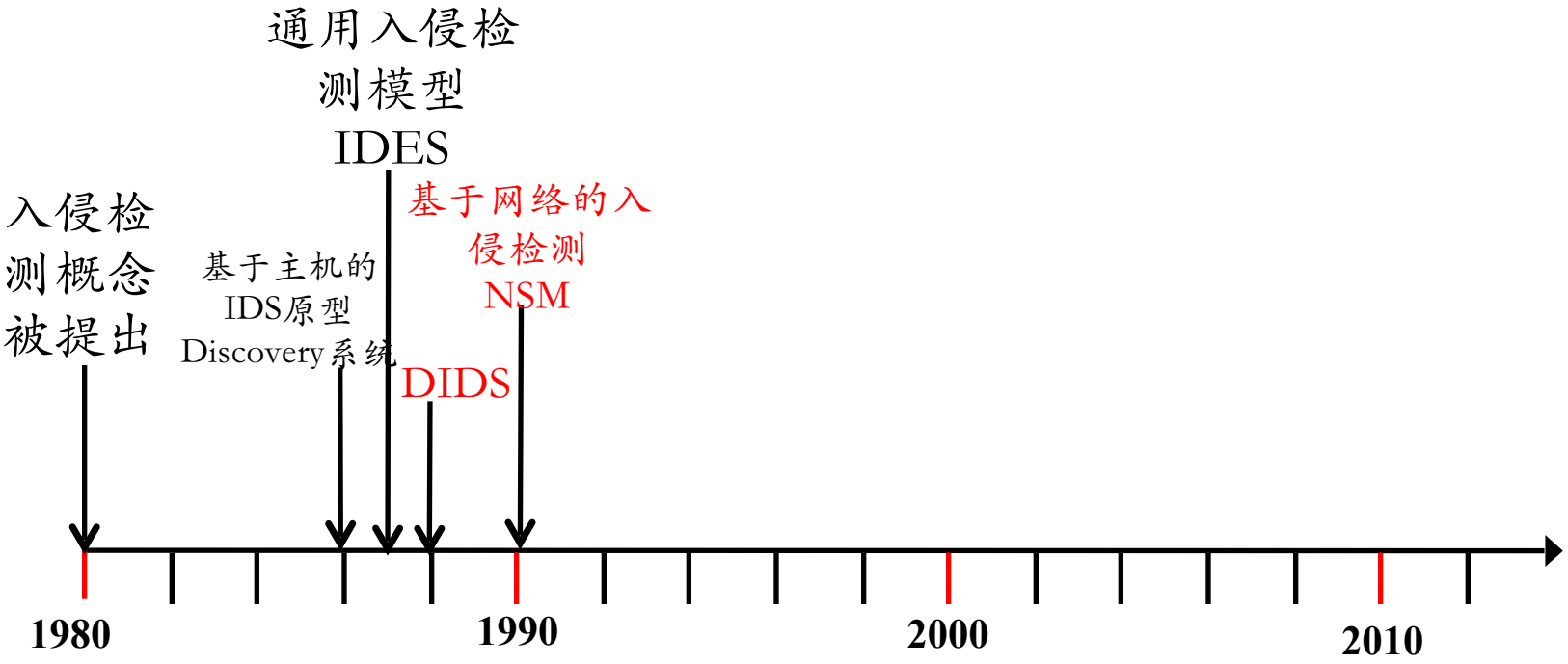
- 1980年 Anderson提出：入侵检测概念，分类方法
- 1986年 W.T.Tener在IBM主机上用COBOL开发了Discovery系统
  —最早的基于主机的IDS原型

入侵检测发展简史（2/5）



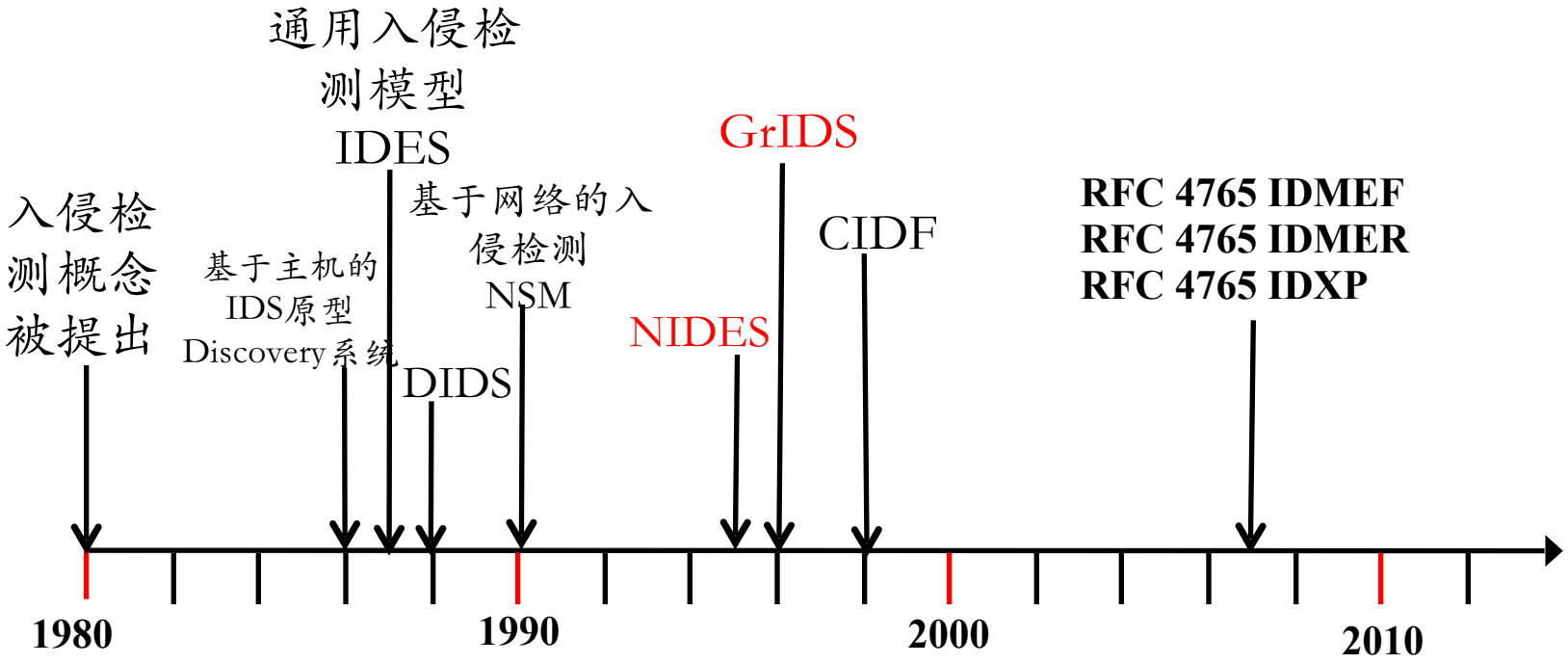
- 1987年 Denning提出了一种通用的入侵检测模型
- IDES: Intrusion Detection Expert System
  - —独立性 ： 系统、环境、脆弱性、入侵种类
  - —系统框架：主体、对象、审计记录、轮廓特征、异常记录、活动规则

入侵检测发展简史（3/5）



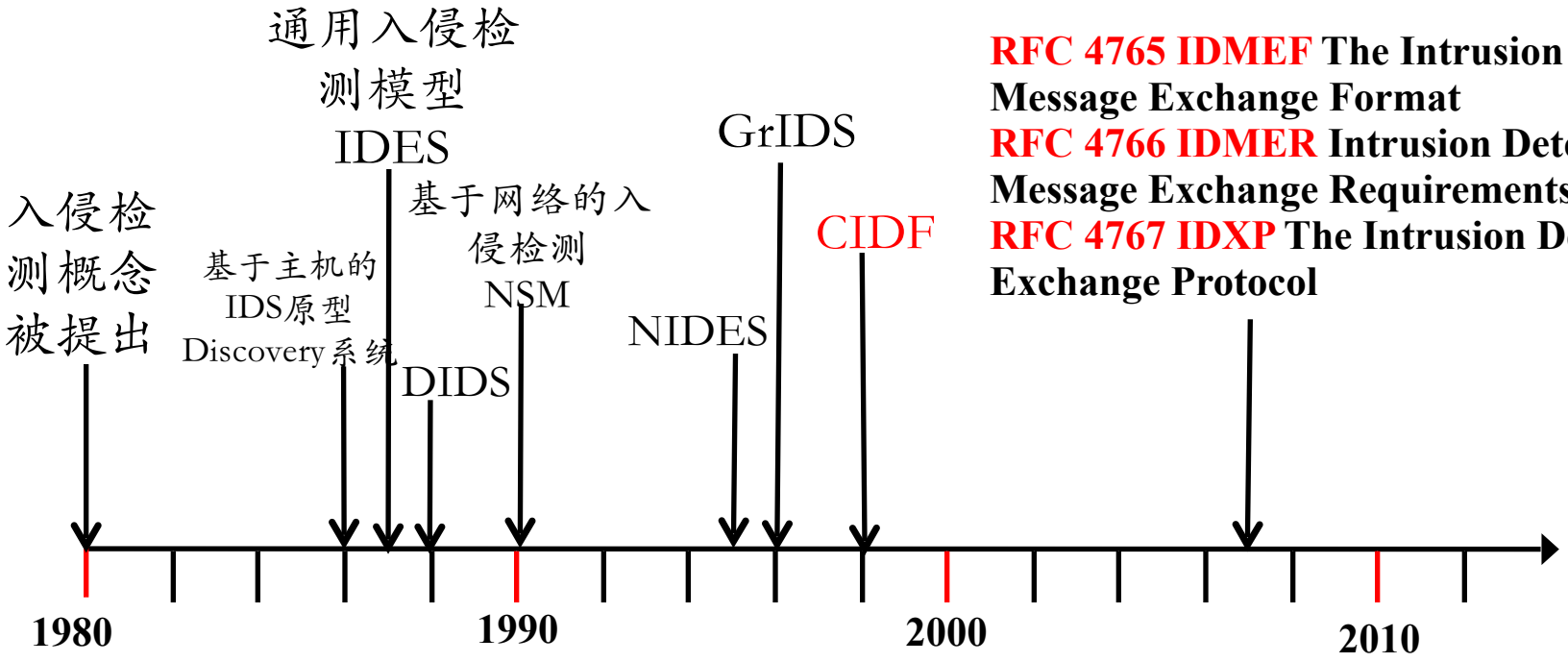
- 1988年 分布式入侵检测，分层体系结构
  —DIDS: Distributed Intrusion Detection System
- 1990年 L.T.Heberlein等提出基于网络的入侵检测NSM
  —Network Security Monitor

入侵检测发展简史（4/5）



- 1995年 Next-Generation IDES
  —检测多个主机上的入侵
- 1996年 GrIDS，针对大规模协同、自动化检测
  —Graph-based Intrusion Detection System

入侵检测发展简史（5/5）



- 1998年 S.Staniford等人提出CIDF
  —Common Intrusion Detection Framework
- 2007年 CIDF工作组的3篇IETF RFC标准草稿发布

入侵检测的必要性

- 实践难：建立安全系统无可能
- 现状难：已安装的缺陷系统的投资保护
- 加密难：加密技术方法本身存在问题
- 管理难：内部用户的滥用
- 效率低：访问控制等级越高使用效率越低
- 模型难：访问控制和保护模型自身存在问题
- 产品难：测试不足、软件生命周期缩短等

入侵检测的作用和意义

- 作用
  - —识别入侵者
  - —识别入侵行为
  - —检测和监视已成功的安全突破
  - —为对抗入侵及时提供重要信息，阻止事件的发生和事态的扩大

防火墙

入侵响应

入侵检测

- 意义
  - —防火墙的有力补充
    - 不是为了预防，是为了检测和提供响应决策
  - —及时、准确、全面的发现入侵

入侵检测技术的发展趋势

- 提高处理能力
  - —10GB、100GB级别网络流量
- 增强协作能力
  - —协作采集 / 协作分析 / 协作响应
  - —针对大规模、协同化网络攻击的趋势
- 丰富检测范围
  - —检测更高级、更隐蔽、更多样化的应用层协议和网络攻击行为
- 标准化
  - —协议 / 产品 / 测试等多个方面

本章内容提要

- 入侵检测发展史
- 入侵检测理论
- 入侵检测关键技术
- 入侵检测标准化
- 入侵检测系统配置

术语定义

- 入侵检测(Intrusion Detection)
- 警报(Alert)
- 特征(Signature)

  —签名

入侵检测

- 对系统的运行状态进行监视，发现各种攻击企图、攻击行为或者攻击结果，以保证系统资源的机密性、完整性和可用性
- 进行入侵检测的软件与硬件的组合便是入侵检测系统

  —IDS : Intrusion Detection System

警报

- 当一个入侵正在发生或者试图发生时，IDS将发布一个alert信息通知系统管理员
- 如果控制台与IDS同在一台机器，alert信息将显示在监视器上，也可能伴随着声音提示
- 如果是远程控制台，那么alert将通过IDS内置方法（通常是加密的）、SNMP（简单网络管理协议，通常不加密）、email、SMS（短信息）或者以上几种方法的混合方式传递给管理员

特征

- IDS的核心是攻击特征（签名），它使IDS在事件发生时触发
- 特征信息过短会经常触发IDS，导致误报或错报，过长则会减慢IDS的工作速度
- 有人将IDS所支持的特征数视为IDS好坏的标准，但是有的厂商用一个特征涵盖许多攻击，而有些产商则会将这些特征单独列出，这就会给人一种印象，好像它包含了更多的特征，是更好的IDS

入侵检测技术分类

- 检测模型
  - —异常检测
  - —误用检测
  - —以上两种模型的融合
- 体系架构
  - —主机入侵检测
  - —网络入侵检测
    - – 集中式
    - – 分布式
- 检测时效性
  - —在线检测
  - —离线检测

入侵检测模型——异常检测

- 用定量方式描述系统可接受的行为特征，用以区分

  —正常用户行为和潜在的入侵行为

- 1980年 Anderson提出的威胁模型

  —外部闯入：未经授权用户的访问

  —内部渗透：已授权用户访问未经授权的数据

  —不当行为：已授权用户对授权数据和资源的不合法或滥用行为

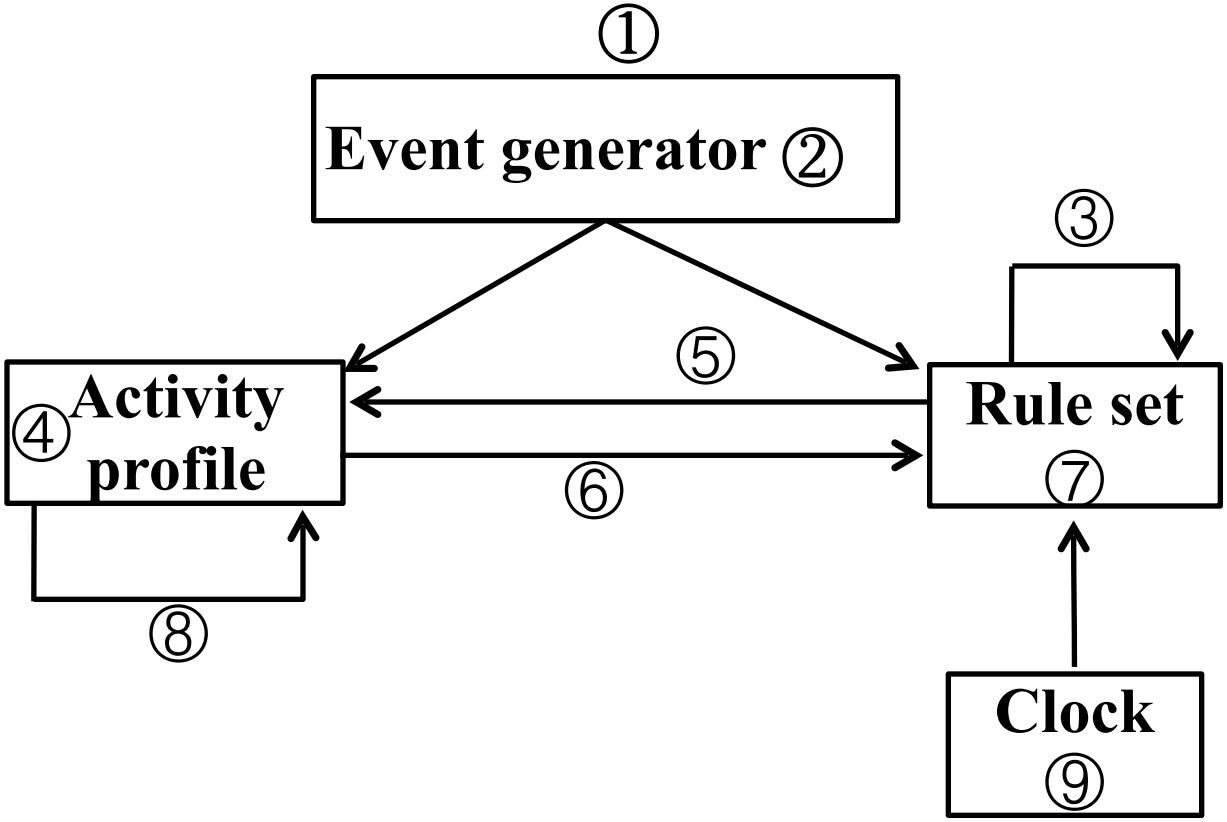
入侵检测模型——误用检测

- 利用已知系统和应用软件的弱点攻击模式来检测入侵
  - —检测已知的漏洞利用行为
  - —无法检测未知的漏洞利用行为

通用入侵检测模型by Dorothy Denning

- 入侵检测技术及系统体系的奠基之作

  —新技术和体系都是在此基础之上的扩展和细化



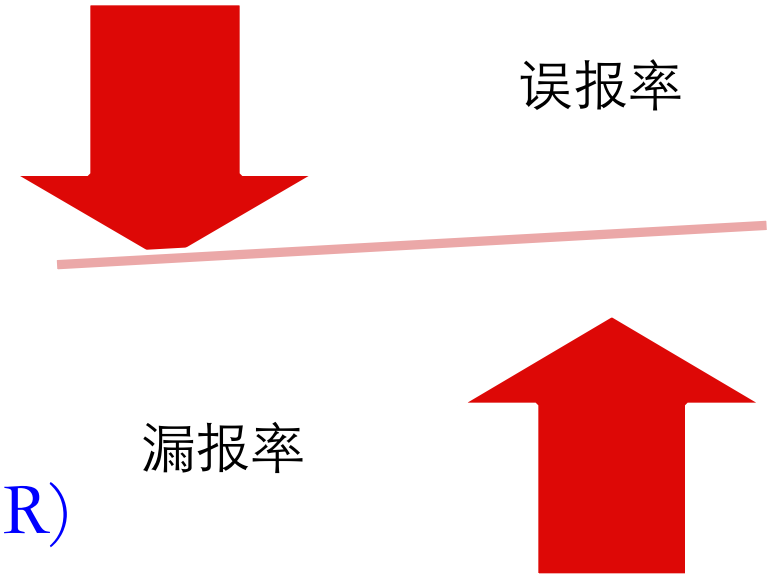
① 审计记录/网络数据包/应用程序记录
② 事件产生器
③ 增加/修改规则
④ 活动记录
⑤ 更新活动记录
⑥ 产生异常活动记录
⑦ 规则集
⑧ 动态产生活动记录
⑨ 时钟

入侵检测技术的评价指标

- 可靠性：容错能力和可持续运行
- 可用性：系统开销要小，对网络性能影响小
- 可测试：通过攻击检测
- 适应性：易于开发、易于扩展
- 实时性：尽早发现入侵企图
- 准确性：
  - 误报率
  - 漏报率
- 安全性：确保自身系统的安全

入侵检测的准确性评价指标

- 误报率
  - —False Positive Rate (FPR)
  - —警报总数：N
  - —误报事件总数：FP
  - —FPR = FP / N
- 漏报率
  - —False Negative Rate (FNR)
  - —FNR = FN / N
- 最理想的入侵检测系统：FPR=0 && FNR=0



两种入侵检测技术的比较

	异常检测	误用检测
可靠性	不稳定	较稳定
可用性	较低	较高
可测试	测试结果波动较大	测试结果较稳定
适应性	较强	较弱
实时性	易造成延时	延时较小
准确性	误报率较高	漏报率较高
安全性	-	-

本章内容提要

- 入侵检测发展史
- 入侵检测理论
- 入侵检测关键技术
- 入侵检测标准化
- 入侵检测系统配置

再看Dorothy Denning的通用入侵检测模型

- 审计记录/网络数据包/应用程序记录
- 事件产生器
- 增加/修改规则
- 活动记录
- 更新活动记录
- 产生异常活动记录
- 规则集
- 动态产生活动记录
- 时钟

- 信息收集
- 信息分析
- 结果处理

信息收集——信息来源

- 系统或网络的日志文件
  —参考《第二章 访问控制》
- 网络流量
  —参考《第四章 网络监听》
- 系统目录和文件的变化
  —参考《第二章 访问控制》
- 程序执行行为
  —资源加载 /API调用序列 / IO规律 …

信息收集——方法

- 基于主机
- 基于网络
- 基于传感器
  - —基于主机运行的软件
  - —基于网络的数据捕获传感器
  - —物联网中的各种传感器

- 异常检测
  —例如：统计分析、完整性分析
- 误用检测
  —例如：模式匹配
- 融合使用异常检测和误用检测
  —实际系统的普遍做法

异常检测之统计分析

- 统计分析对象

  —如用户、文件、目录和设备等

- 统计分析方法

  —为统计分析对象创建一个统计描述，统计正常使用时的一些测量属性

  – 如访问次数、操作失败次数和延时等

- 统计匹配

  —测量属性的平均值将被用来与网络、系统的行为进行比较，任何观测/测量值在正常值范围之外时，就认为有入侵发生

异常检测之完整性分析

- 完整性分析对象
  - —文件 / 目录 / 任意数字资源
    - 例如：文件和目录的内容及属性
- 完整性分析方法
  - —建立完整性分析对象在正常状态时的完整性签名
- 完整性分析匹配
  - —匹配签名值是否发生改变
    - 若发生改变，则认定目标对象被入侵篡改

其他常见异常检测算法

- 基于特征选择异常检测
- 基于贝叶斯推理异常检测
- 基于贝叶斯网络异常检测
- 基于模式预测异常检测
- 基于神经网络异常检测
- 基于贝叶斯聚类异常检测
- 基于机器学习异常检测
- 基于数据挖掘异常检测

误用检测之模式匹配

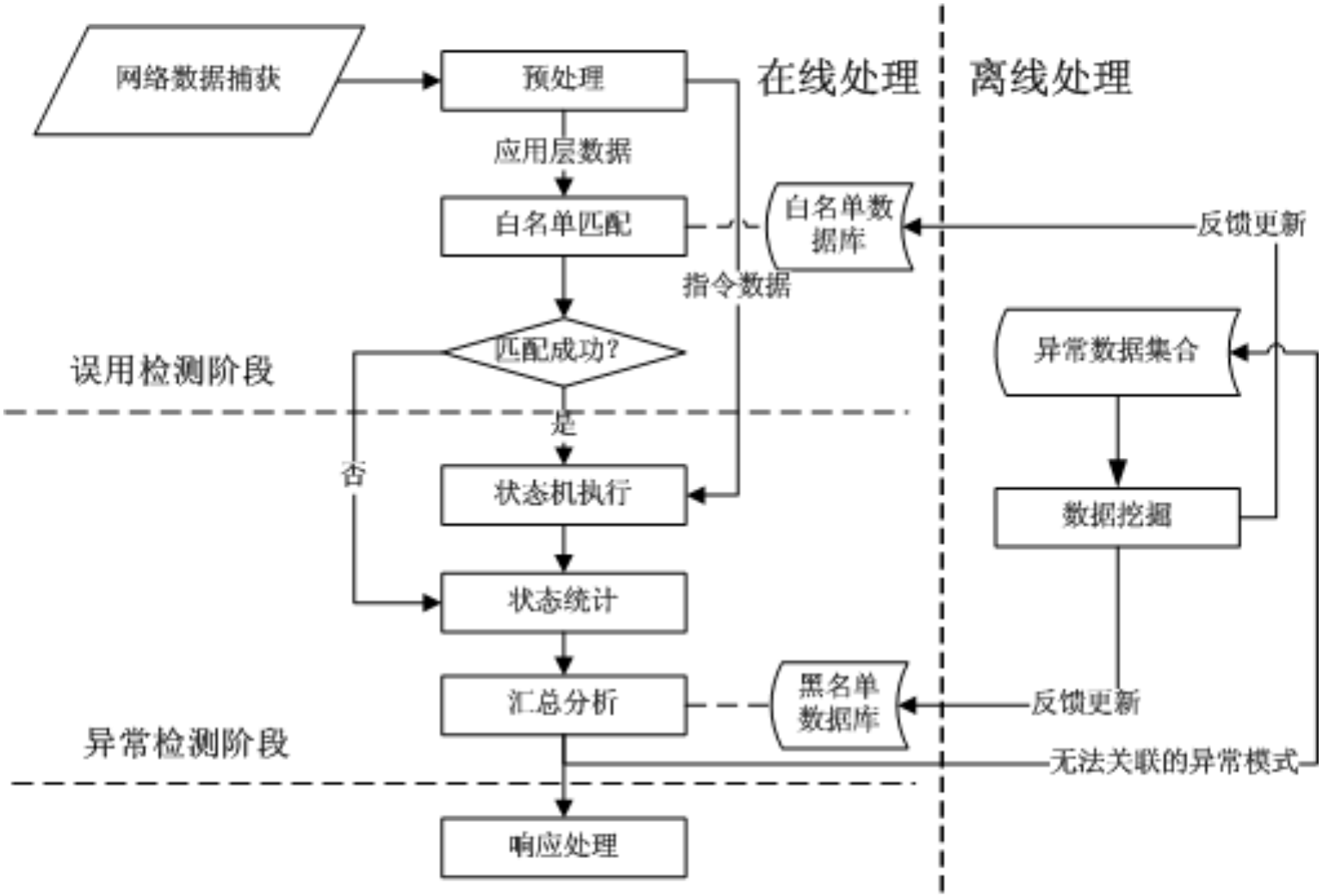
- 模式匹配就是将收集到的信息与已知的网络入侵和系统误用模式规则集进行比较，从而发现违反安全策略的行为
- 入侵模式的表示方法
  - ——一个过程（如执行一条指令）
  - ——一个输出（如获得权限）
- 入侵模式的匹配过程
  - ——字符串匹配：精确匹配、模糊匹配
  - ——状态机迁移序列匹配

其他常见误用检测算法

- 基于条件概率误用检测
- 基于专家系统误用检测
- 基于状态迁移误用检测
- 基于模型误用检测

融合使用异常检测和误用检测

- 示例



结果处理

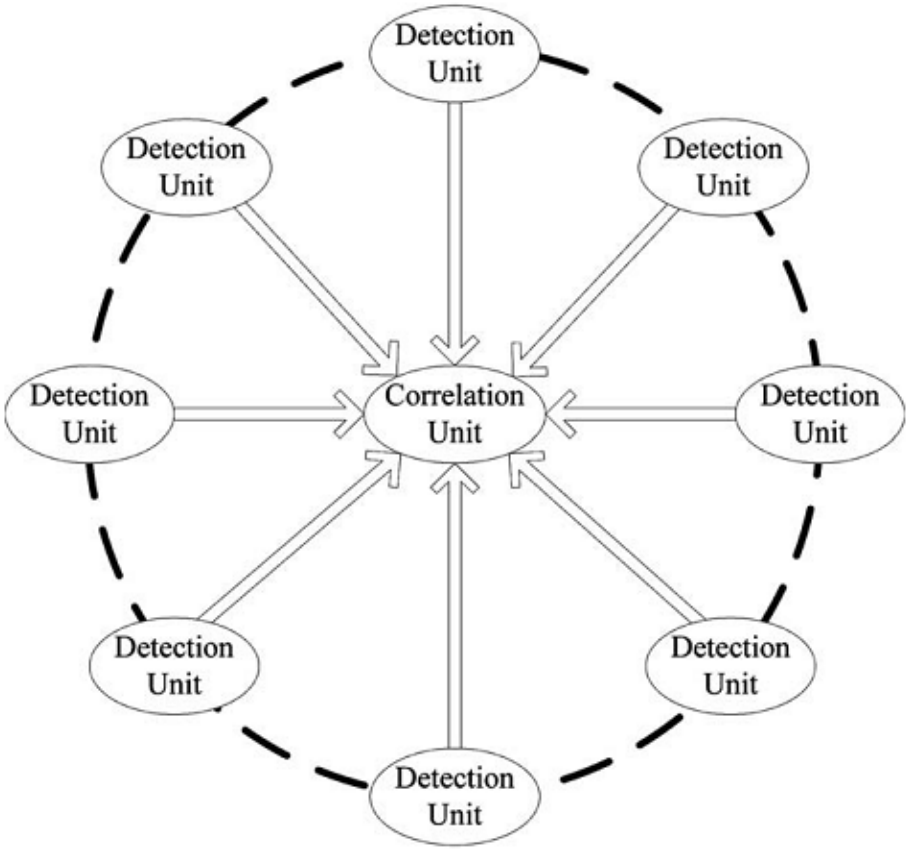
- 产生告警
  - —记录告警日志
  - —请求其他安全设备的协作联动
    - 防火墙联动

入侵检测体系架构

- 主机入侵检测系统
  - —监视与分析主机的审计记录
    - 能否及时采集到审计记录
  - —可以不运行在监控主机上
    - 如何保护作为攻击目标的主机审计子系统?
- 网络入侵检测系统
  - —对通信数据进行侦听采集数据
  - —提供对网络通用的保护
    - 不消耗主机额外资源
    - 如何适应高速网络环境?

网络入侵检测体系架构——集中式(1/2)

- 分布式信息收集
- 汇总上报
- 集中关联分析
- 典型代表
  - —DIDS (Snapp et al., 1991)
  - —DShield (Internet Storm Center)
  - —NSTAT (Kemmerer, 1998)
    - Heberlein et al., 1992;
    - Hochberg et al.,1993;
    - Mounji et al., 1995;
    - Huang et al., 1999

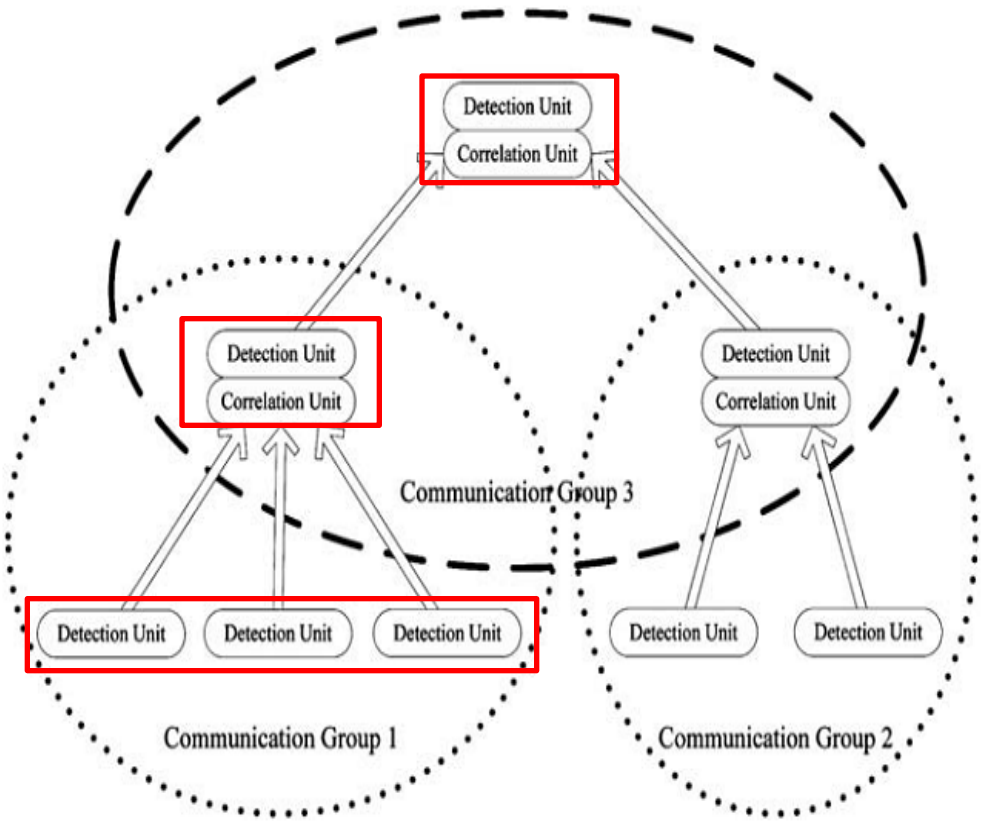


网络入侵检测体系架构——集中式(2/2)

- 缺点
  - —单点故障
    - 汇总关联分析节点一旦出现故障，则整个系统就会无法工作
  - —单点瓶颈
    - 汇总关联分析节点的处理能力决定了整个系统的处理能力
- 应用场景
  - —小型公司网络范围内的IDS协作
  - —不适合于互联网范围内的大规模IDS协作（通信延时）

网络入侵检测体系架构——等级分布式(1/3)

- 特点
  - —分区
    - – 地理位置
    - – 管理域
    - – 相近软件平台
    - – 预期入侵种类
  - —“逐层”汇聚式关联分析



网络入侵检测体系架构——等级分布式(2/3)

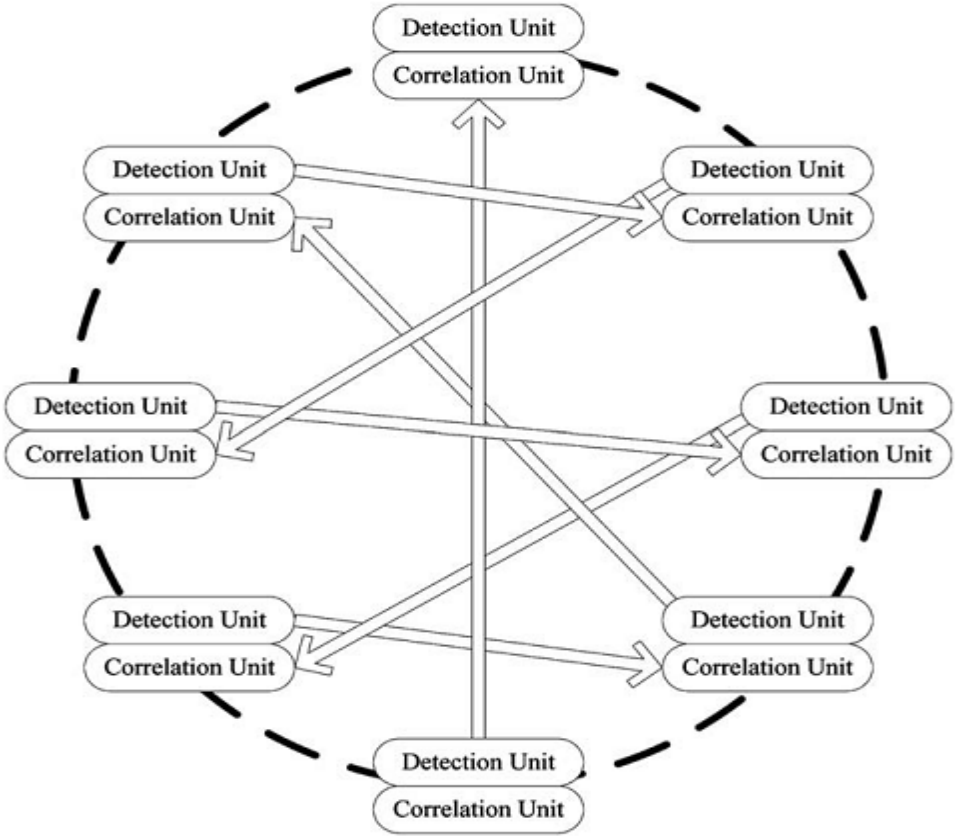
- 典型代表
  - —GrIDS (Staniford-Chen et al., 1996)
  - —The EMERALD project (Porras and Neumann, 1997)
  - —Li et al. (2007)
    - a hierarchical CIDS based on dependency
  - —DSOC (Distributed Security Operation Center, Abdoul Karim Ganame et al., 2008)
  - —Servin and Kudenko (2008)
    - reinforcement learning based
  - —AAFID(Balasubramaniyan et al., 1998)
  - —NetSTAT (Vigna, 1999)

网络入侵检测体系架构——等级分布式(3/3)

- 缺点
  - —可扩展性略高于集中式架构，但仍受困于高层次节点能力
  - —单点故障仍有可能（高层节点）
  - —检出率较低
    - 信息在“汇聚”的过程中会由于“压缩”而导致“损失”和“失真”

网络入侵检测体系架构——完全分布式(1/3)

- 特点
  - —无超级管理节点
  - —节点间通信
    - – P2P
    - – Gossip协议
    - – 组播
    - – 发布/订阅机制



网络入侵检测体系架构——完全分布式(2/3)

- 典型代表
  - —Locasto et al. (2005): P2P based
  - —The DOMINO project (Yegneswaran et al., 2004; Barford and Jha, 2004)
  - —Dash et al. (2006)
  - —Garcia et al. (2004)
  - —MADIDF (Mobile Agents based Distributed Intrusion Detection Framework) (Dayong Ye et al., 2008)
  - —Indra (Janakiraman and Zhang, 2003)
  - —CSM (White et al., 1996)

网络入侵检测体系架构——完全分布式(3/3)

- 缺点
  - —检测精度低
  - —可扩展性差
  - —负载均衡难度高

协作式攻击与分布式检测的流程(1/2)

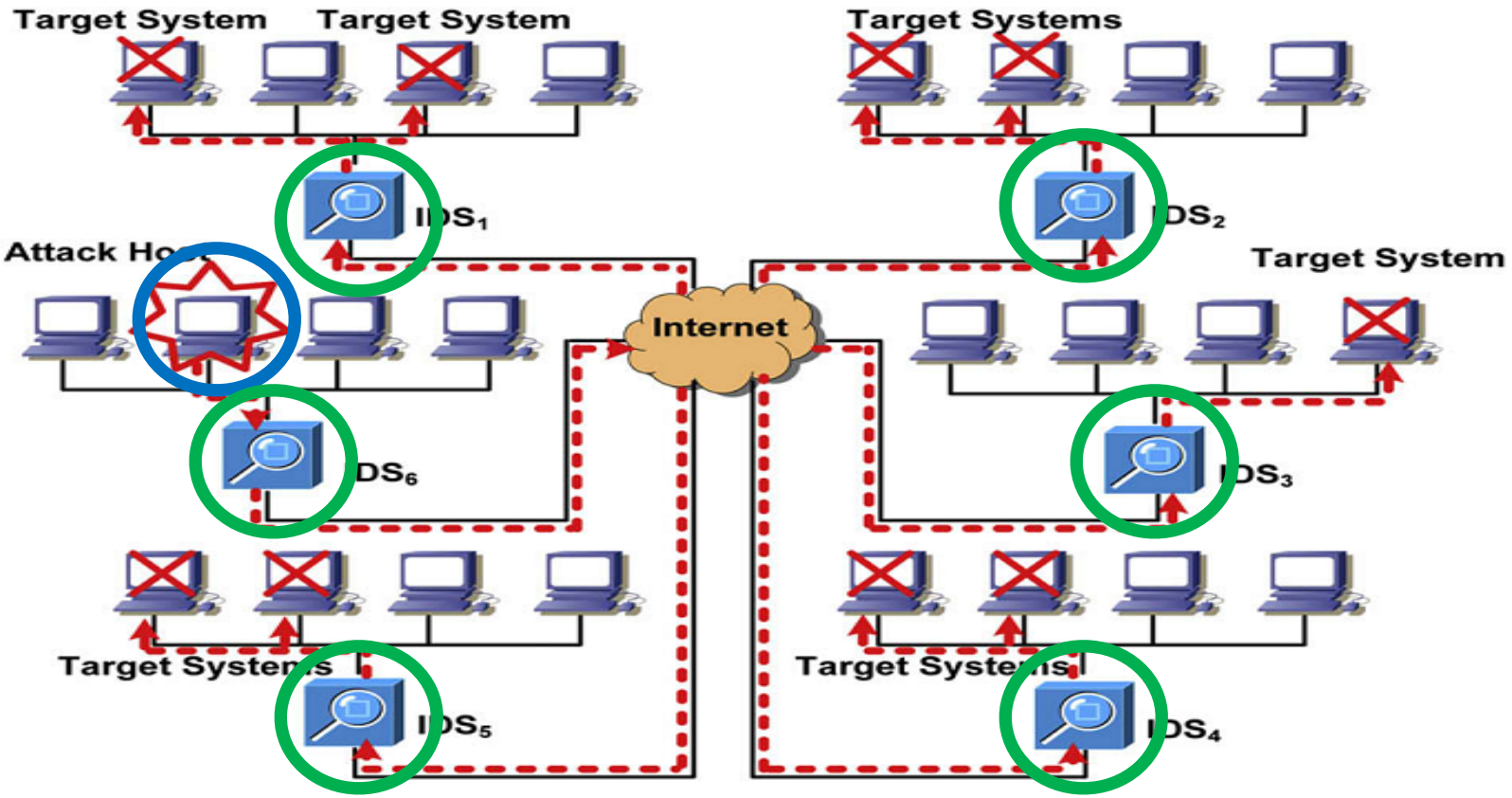


Fig. 1 – A common stage of coordinated attacks.

协作式攻击与分布式检测的流程(2/2)

- 协作式攻击的特点
  - —分布式
  - —协同
  - —有发起源
- 分布式检测的优势
  - —快速定位协作式攻击的真正源头
  - —避免对同源攻击的重复报警

本章内容提要

- 入侵检测发展史
- 入侵检测理论
- 入侵检测关键技术
- 入侵检测标准化
- 入侵检测系统配置

通用入侵检测框架——CIDF概述

- 美国国防高级研究项目局（DARPA）为IDS数据交换而作出的一个尝试
  - —RFC 4765 The Intrusion Detection Message Exchange Format
  - —RFC 4766 Intrusion Detection Message Exchange Requirements
  - —RFC 4767 The Intrusion Detection Exchange Protocol
- 定义了IDS表达检测信息的标准语言以及IDS组件之间的通信协议
  - —符合CIDF规范的IDS和安全设备可以共享检测信息，协同工作
- 集成各种安全设备使之协同工作
  - —分布式入侵检测的基础

通用入侵检测框架——CIDF文档组成

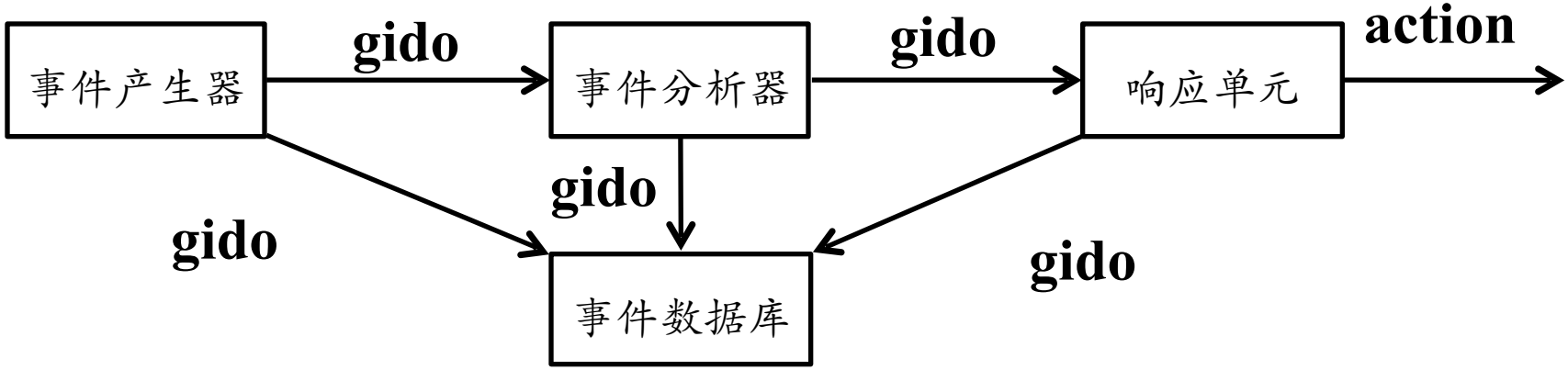
- 体系结构
  - —提出了一个标准的IDS的通用模型
- 规范语言
  - —CISL: A Common Intrusion Specification Language
  - —定义了一个用来描述各种检测信息的标准语言
- 内部通讯
  - —定义了IDS组件之间进行通信的标准协议
- 程序接口
  - —提供了一整套标准的应用程序接口（API函数）

通用入侵检测框架——体系结构(1/4)

- 事件
  - —IDS需要分析的数据
  - —基于网络的IDS从网络中提取的数据包
  - —基于主机的IDS从系统日志等其它途径得到的数据信息
- CIDF组件之间的交互数据格式定义
  - —通用入侵检测对象

通用入侵检测框架——体系结构(2/4)

- 通用入侵检测对象
  - —generalized intrusion detection objects (gido)
  - —表示在一些特定时刻发生的一些特定事件
  - —表示从一系列事件中得出的一些结论
  - —表示执行某个行动的指令



通用入侵检测框架——体系结构(3/4)

- 事件产生器
  - —数据采集
  - —从入侵检测系统外的整个计算环境中获得事件，并以CIDF gido格式向系统的其他部分提供此事件
  - —事件产生器是所有IDS所需要的，同时也是可以重用的
- 事件分析器
  - —分析部分
  - —从其他组件接收gido，分析得到的数据，并产生新的gido
  - —误用/异常检测算法均可以在此得到应用

通用入侵检测框架——体系结构(4/4)

- 响应单元
  - —响应部分
  - —是对分析结果作出作出反应的功能单元，它可以终止进程、重置连接、改变文件属性等，也可以只是简单的报警
  - —也可以和防火墙等其他安全设备联动响应
- 事件数据库
  - —日志
  - —是存放各种中间和最终数据的地方的统称，它可以是复杂的数据库，也可以是简单的文本文件
  - —持久化存储

通用入侵检测框架——规范语言

- 各IDS使用统一的CISL来表示
  - —原始事件信息（审计踪迹记录和网络数据流信息）
  - —分析结果（系统异常和攻击特征描述）
  - —响应指令（停止某些特定的活动或修改组件的安全参数）
  - —建立了IDS之间信息共享的基础
- CISL是CIDF的最核心也是最重要的内容
- GIDO的构建与编码是CISL的重点

通用入侵检测框架——内部通讯

- 内部通讯机制之匹配服务
  - —为CIDF各组件之间的相互识别、定位和信息共享提供了一个标准的统一的机制
  - —基于LDAP协议实现
- 内部通讯机制之消息层
  - —在易受攻击的环境中实现了一种安全（保密、可信、完整）并可靠的信息交换机制
    - 使通信与阻塞和非阻塞处理无关
    - 使通信与数据格式无关
    - 使通信与操作系统无关
    - 使通信与编程语言无关

通用入侵检测框架——程序接口

- 程序员可以在不了解编码和传递过程具体细节的情况下，以一种很简单的方式构建和传递GIDO
  - —GIDO编解码API
  - —消息层API
  - —GIDO传输API
  - —GIDO动态追加API
  - —签名API
  - —顶层CIDF的API

OPSEC: Open Platform for Security

- OPSEC 联盟成立于1997年
  - —发起者：Checkpoint
  - —成立目的：向用户提供完整的、能够在多厂商之间进行紧密集成的网络安全解决方案
  - —联盟组成
    - 提供集成的应用程序
    - 提供基于Checkpoint平台的安全服务

OPSEC体系概述(1/3)

- 内容安全
  - —CVP (Content Vectoring Protocol)
- Web资源管理
  - —UFP (URL Filtering Protocol)
- 入侵检测
  - —SAM (Suspicious Activity Monitoring)
- 事件集成
  - —Log Export API
  - —Event Logging API

OPSEC体系概述(2/3)

- 管理和分析
  - —CPMI (Check Point Management Interface)
  - —AMON (Application Monitoring)
- 认证
  - —Secure Authentication API (SAA)
- 带负载平衡的高可用性和双机热备份
  - —Load Balancing (HA/LB)
  - —Hot Standby (HA - HS)
- 用户到地址的映射
  - —UAM (User Address Mapping)

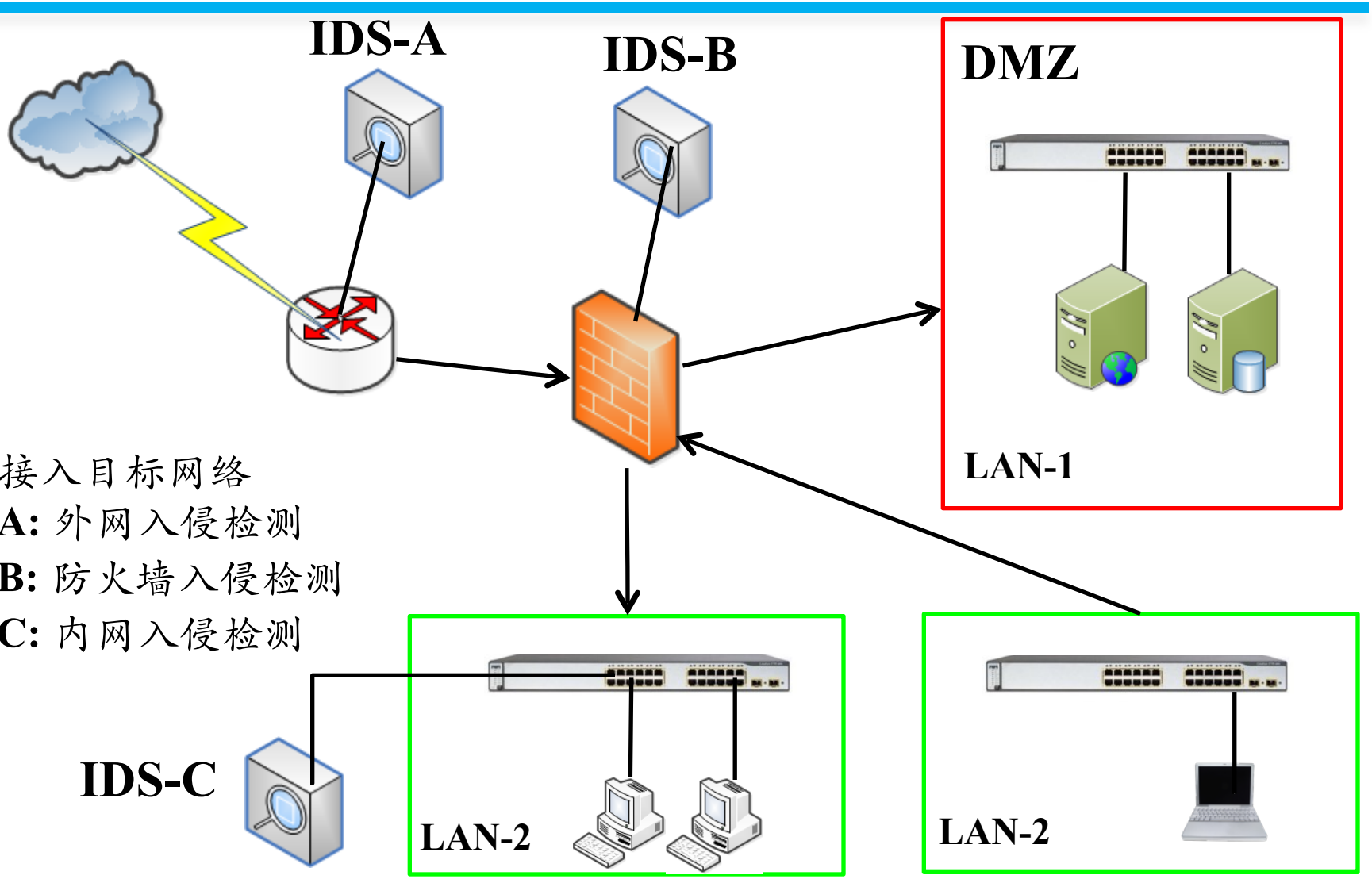
OPSEC体系概述(3/3)

- 电子商务安全
  —UserAuthorityT API (UAA)
- 工业标准协议
  —Radius/TACACS+ （认证协议）
  —SNMP （简单网络管理协议）
  —LDAP （轻量目录访问协议）

本章内容提要

- 入侵检测发展史
- 入侵检测理论
- 入侵检测关键技术
- 入侵检测标准化
- 入侵检测系统配置

入侵检测系统的典型部署模式



入侵检测系统的开源代表

- Snort的产品定位
  - —多模式报文分析工具
    - 嗅探工具
    - 报文记录
    - 数据取证分析工具
    - 网络入侵检测系统(NIDS)
- 应用场景
  - —在线分析
  - —离线分析

Snort的特性列表

- 轻巧
  —源代码不到1MB
- 移植性
  —Linux, Windows, MacOS X, Solaris, BSD, IRIX, Tru64, HP-UX
- 高性能
  —百兆网络线速处理
- 可配置
  —简单的规则语言，丰富的日志/配置分析工具
- 免费
  —GPL开源

Snort的设计理念

- 轻量级网络入侵检测系统
- 基于Libpcap的报文嗅探接口
- 基于规则的检测引擎
- 插件系统支持无限扩展

检测引擎

- 基于指纹的规则
- 模块化设计
- 丰富的检测能力（内置规则）
  —隐蔽扫描，操作系统识别扫描，缓冲区溢出攻击，后门，CGI漏洞利用，等等
- 灵活的规则系统设计
  —易于创建新规则

Snort的插件

- 预处理器
  —报文在被送到检测引擎之前可以先进行预处理
- 检测
  —针对单个报文/报文字段的快速检测
- 输出
  —获取其他插件的输出结果

使用Snort

- 三种主要的工作模式
  - —嗅探模式
  - —报文记录模式
  - —网络入侵检测系统(NIDS)
  - —(数据取证分析工具)
- 通过命令行选项配置不同的工作模式
  - —默认以NIDS模式启动

使用Snort – 嗅探模式

- 类似tcpdump
- 解码报文后输出到标准输出(控制台)
- 支持BPF过滤规则

捕获的数据报文截图

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

11/09-11:12:02.954779 10.1.1.6:1032 -> 10.1.1.8:23
TCP TTL:128 TOS:0x0 ID:31237 IpLen:20 DgmLen:59 DF
***AP*** Seq: 0x16B6DA  Ack: 0x1AF156C2  Win: 0x2217  TcpLen: 20
FF FC 23 FF FC 27 FF FC 24 FF FA 18 00 41 4E 53  ..#..'..$....ANS
49 FF F0                                         I..

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

11/09-11:12:02.956582 10.1.1.8:23 -> 10.1.1.6:1032
TCP TTL:255 TOS:0x0 ID:49900 IpLen:20 DgmLen:61 DF
***AP*** Seq: 0x1AF156C2  Ack: 0x16B6ED  Win: 0x2238  TcpLen: 20
0D 0A 0D 0A 53 75 6E 4F 53 20 35 2E 37 0D 0A 0D  ....SunOS 5.7...
00 0D 0A 0D 00                                   .....

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

报文记录模式

- 多种报文记录模式
  —ASCII, tcpdump, XML, 数据库
- 可以对记录的报文进行后续的分析/取证

NIDS 模式

- BackTrack 5自带的Snort 2.8.5.2 (Build 121)有4000+条规则
- 支持多种检测模式
  - —基于规则
  - —统计异常
  - —基于协议

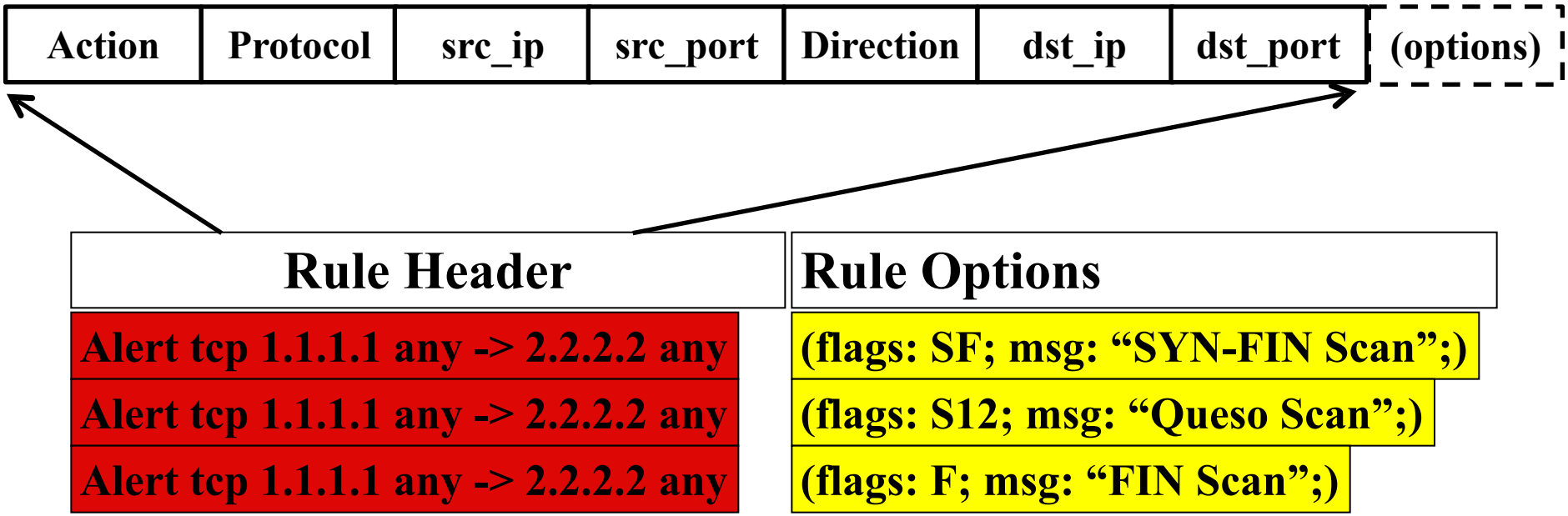
Snort规则(1/3)

```
root@bt:/etc/snort/rules# grep "alert" -R *.rules | wc -l
4072
root@bt:/etc/snort/rules# ls
attack-responses.rules          community-web-dos.rules    policy.rules
backdoor.rules                  community-web-iis.rules    pop2.rules
bad-traffic.rules               community-web-misc.rules   pop3.rules
chat.rules                      community-web-php.rules    porn.rules
community-bot.rules             ddos.rules                 rpc.rules
community-deleted.rules         deleted.rules              rservices.rules
community-dos.rules             dns.rules                  scan.rules
community-exploit.rules         dos.rules                  shellcode.rules
community-ftp.rules             experimental.rules         smtp.rules
community-game.rules            exploit.rules              snmp.rules
community-icmp.rules            finger.rules               sql.rules
community-imap.rules            ftp.rules                  telnet.rules
community-inappropriate.rules   icmp-info.rules            tftp.rules
community-mail-client.rules     icmp.rules                 virus.rules
community-misc.rules            imap.rules                 web-attacks.rules
community-nntp.rules            info.rules                 web-cgi.rules
community-oracle.rules          local.rules                web-client.rules
community-policy.rules          misc.rules                 web-coldfusion.rules
community-sip.rules             multimedia.rules           web-frontpage.rules
community-smtp.rules            mysql.rules                web-iis.rules
community-sql-injection.rules   netbios.rules              web-misc.rules
community-virus.rules           nntp.rules                 web-php.rules
community-web-attacks.rules     oracle.rules               x11.rules
community-web-cgi.rules         other-ids.rules
community-web-client.rules      p2p.rules
```

Snort规则(2/3)

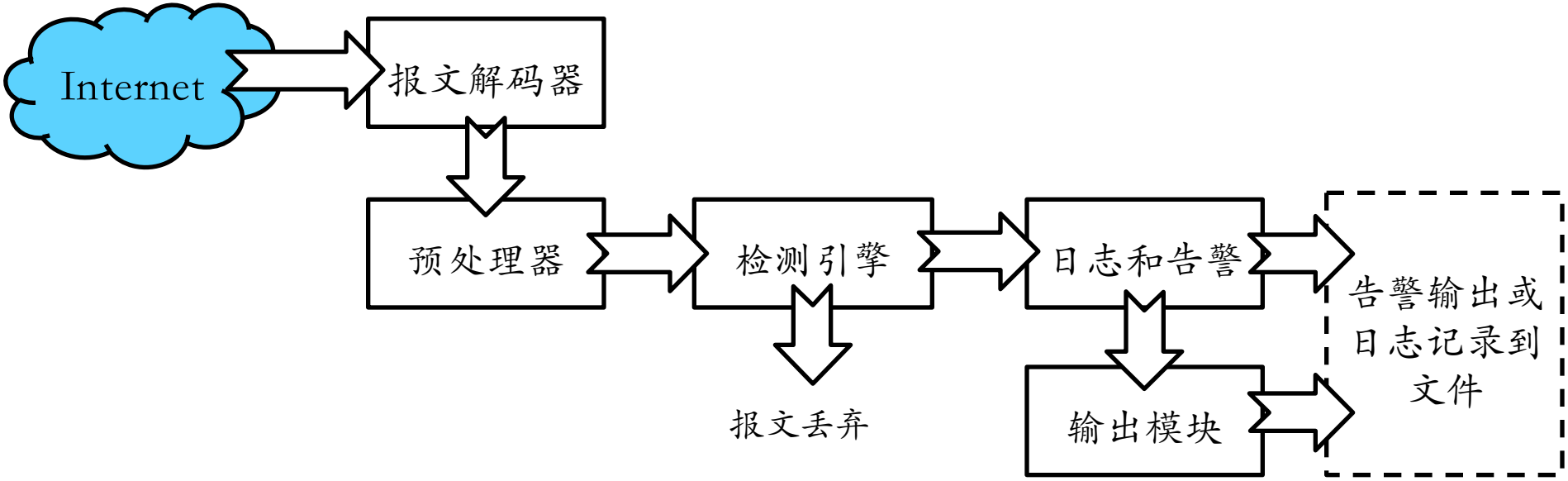
- 单行规则
- snort.conf定义了默认规则文件位置

  —var RULE_PATH /etc/snort/rules



Snort规则(3/3)

- 规则执行顺序
  - —不按照规则出现顺序
  - —按规则类型优先级
    - – Pass > Drop > Alert > Log



Snort规则之误用检测

- 绝大多数snort规则都是误用检测规则
- 识别已知的入侵行为
- Snort的主要计算资源都用在了字符串匹配

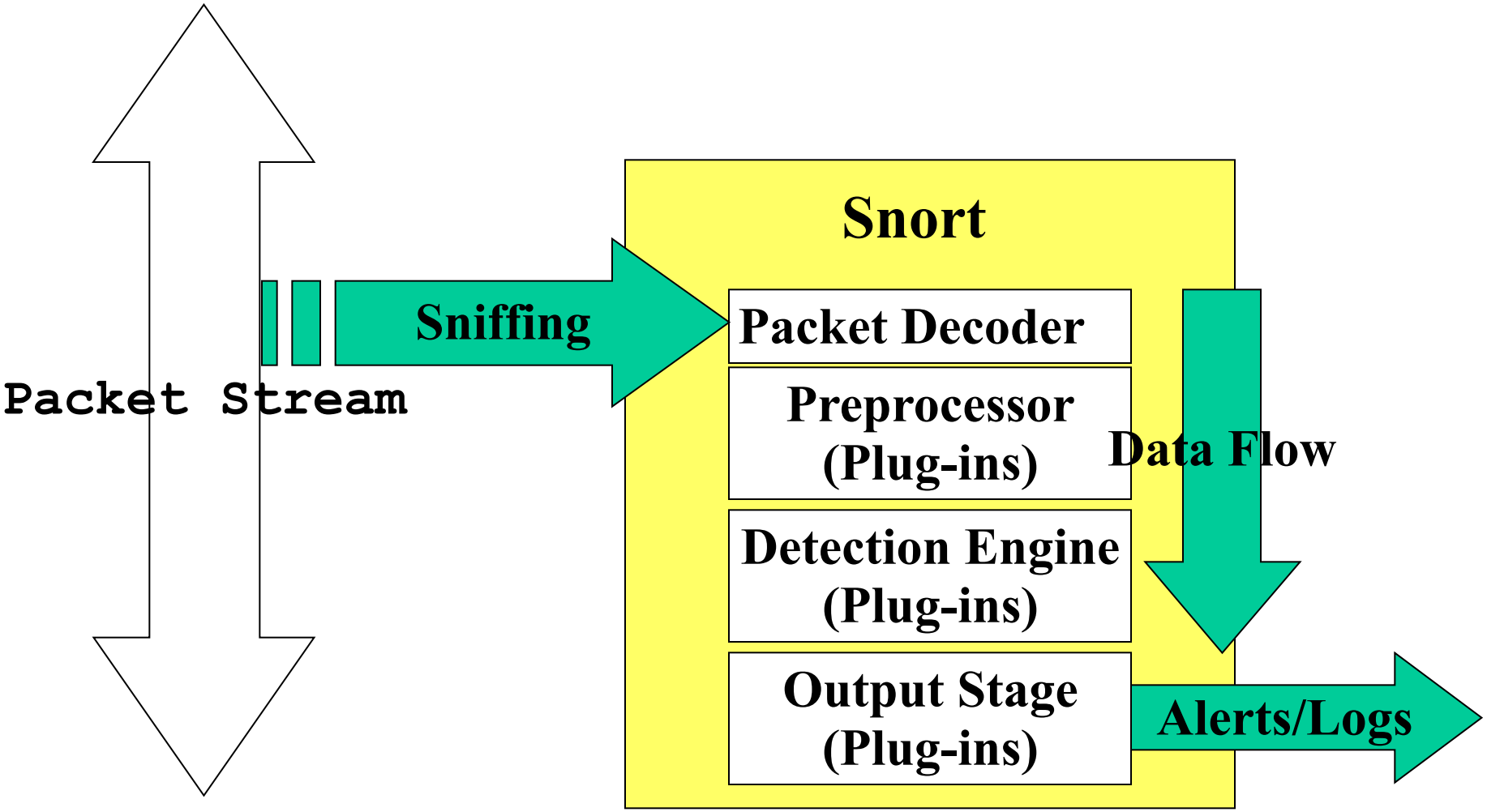
```
alert tcp $EXTERNAL_NET any -> $SQL_SERVERS 3306 (msg:"MYSQL root login attempt"; flow:t
o_server,established; content:"|0A 00 00 01 85 04 00 00 80|root|00|"; classtype:protocol
-command-decode; sid:1775; rev:2;)
alert tcp $EXTERNAL_NET any -> $SQL_SERVERS 3306 (msg:"MYSQL show databases attempt"; fl
ow:to_server,established; content:"|0F 00 00 00 03|show databases"; classtype:protocol-c
ommand-decode; sid:1776; rev:2;)
alert tcp $EXTERNAL_NET any -> $SQL_SERVERS 3306 (msg:"MYSQL 4.0 root login attempt"; fl
ow:to_server,established; content:"|01|"; within:1; distance:3; content:"root|00|"; with
in:5; distance:5; nocase; classtype:protocol-command-decode; sid:3456; rev:2;)
```

Snort规则之异常检测

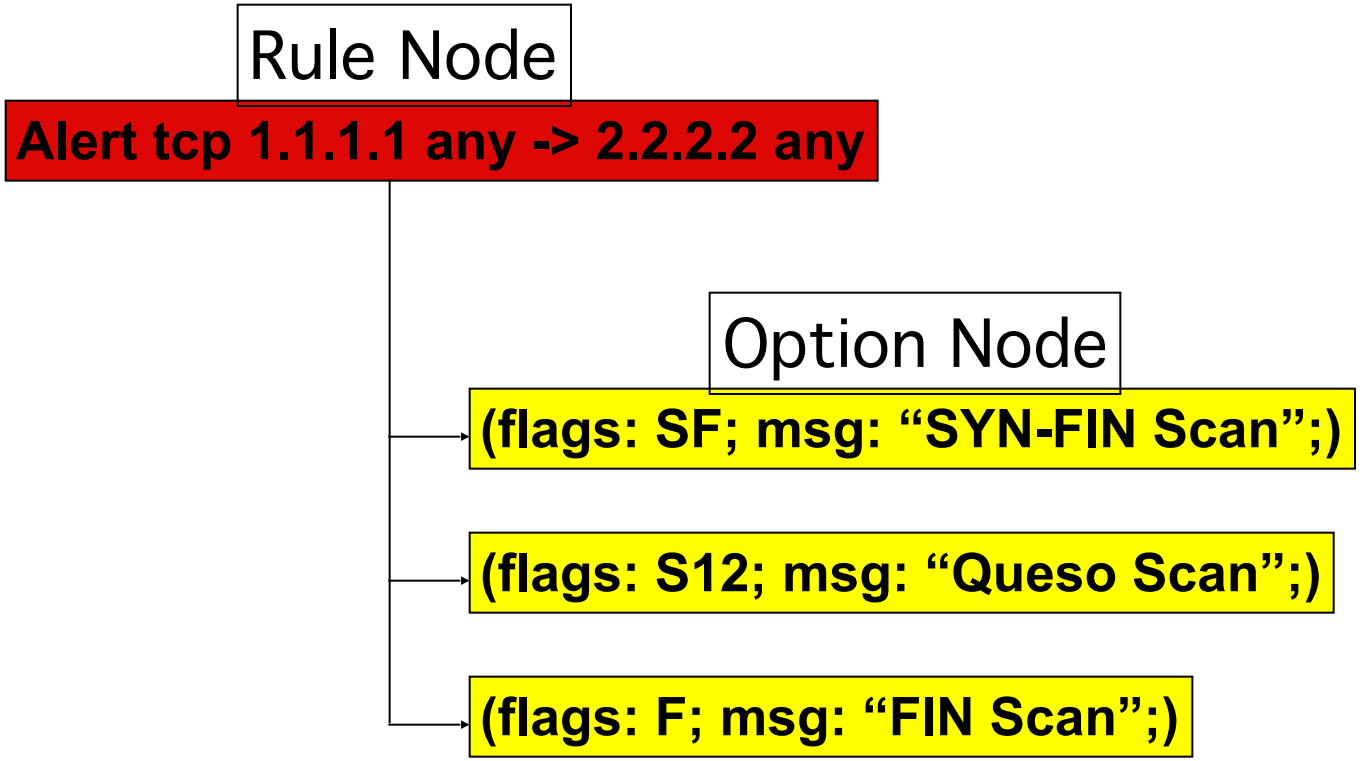
- 极少数的snort规则
- 基于统计异常
  - —threshold.conf
  - —阈值法
    - Limit: 每统计周期的第一次阈值到达时报警
      每统计周期最多报警一次
    - Threshold: 每统计周期的每M次阈值到达时报警
      每统计周报报警次数不限
    - Both: 以上2种的综合

```
alert tcp $EXTERNAL_NET any <> $HOME_NET 179 (msg:"DOS BGP spoofed connection reset atte
mpt"; flow:established; flags:RSF*; threshold:type both,track by_dst,count 10,seconds 10
; reference:bugtraq,10183; reference:cve,2004-0230; reference:url,www.uniras.gov.uk/vuls
/2004/236929/index.htm; classtype:attempted-dos; sid:2523; rev:7;)
```

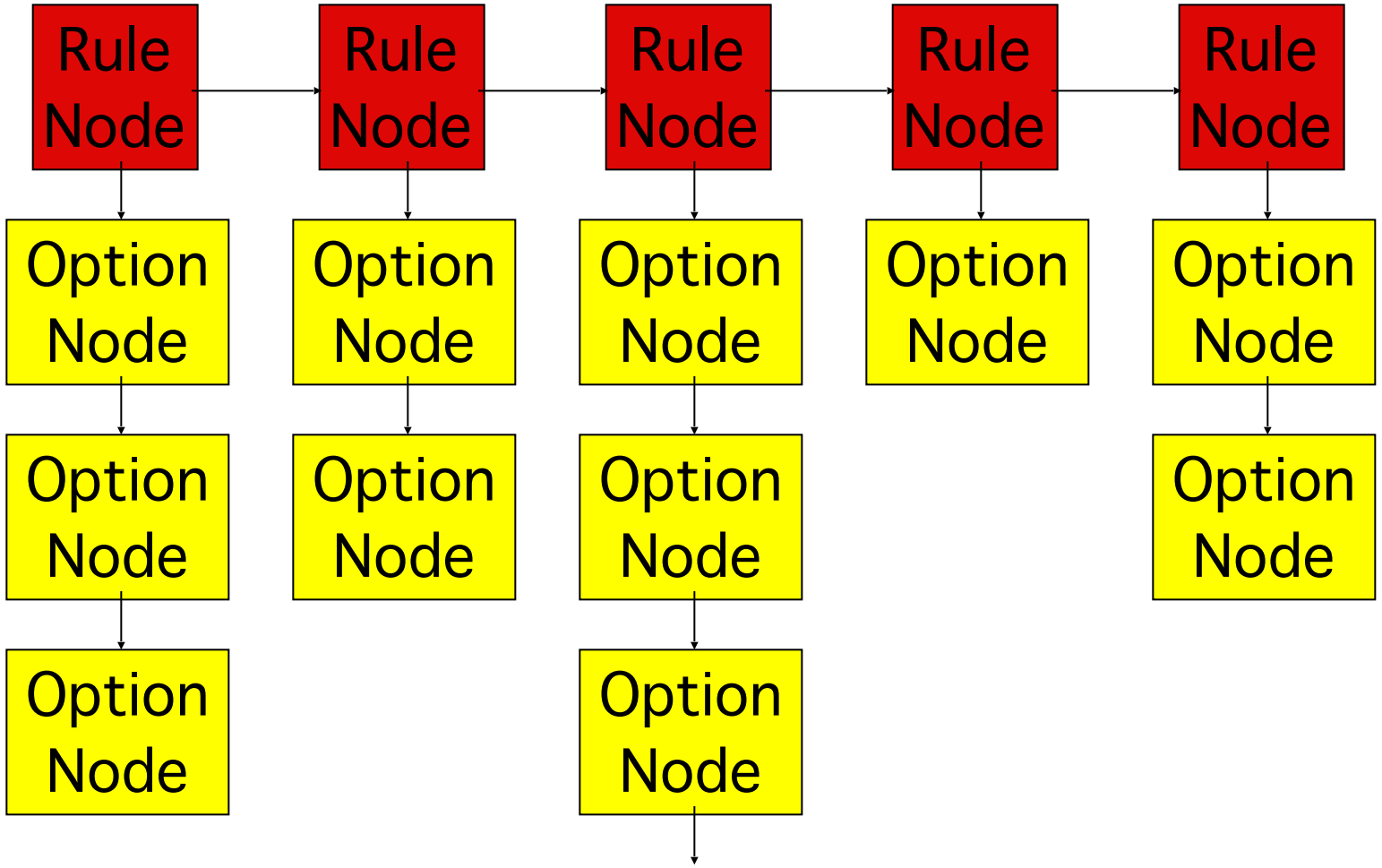
Snort架构之数据流



Snort架构之检测引擎：内部数据结构表示



Snort架构之检测引擎：完全展开



Snort配置

- /etc/snort/snort.conf
  - —定义网络环境相关变量
  - —配置动态加载的库
  - —配置预处理器
  - —配置输出插件
  - —增加运行时配置指令
  - —定义启用的规则集

实验讲解

- 实验一：配置snort为嗅探模式
- 实验二：配置并启用snort内置规则
- 实验三：自定义snort规则
- 实验四：和防火墙联动

实验一： 配置snort为嗅探模式

- 显示IP/TCP/UDP/ICMP头
  —snort – v
- 显示应用层数据
  —snort -vd
- 显示数据链路层报文头
  —snort -vde

实验三：自定义snort规则

- 模拟DoS攻击
  - —nping --tcp-connect -c 256 --rate 512 192.168.56.101 -p 80
- 检测规则
  - —alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"Possible too many connections toward my http server"; threshold:type threshold, track by_src, count 100, seconds 2; classtype:attempted-dos; sid:1000002; rev:1;)

实验四：和防火墙联动

- guardian.pl
  - —guardian.pl -c guardian.conf

参考文献

1. **Intrusion Detection Exchange Format http://datatracker.ietf.org/wg/idwg/**
2. 蒋建春, 马恒太, 任党恩, 卿斯汉 网络安全入侵检测：研究综述.软件学报. **vol. 11. 2000. pp. 1460–1466.**
3. **C.V. Zhou, C. Leckie, and S. Karunasekera, A survey of coordinated attacks and collaborative intrusion detection. Computers & Security. vol. In Press, Corrected Proof. 2009.**
4. **OPSEC: Open Platform for Security http://www.opsec.com/**
5. **snort官方网站 http://www.snort.org/**

课后思考题

- 如何理解“入侵检测系统的误报率越高，漏报率低的概率就越高”这句话?
- 描述入侵检测系统的误用检测算法和异常检测算法，并各举一实际算法说明