高级网络安全研究与应用——

攻击分析与攻击检测

北京邮电大学

郑康锋 zkfbupt@163.com

伍淳华 wuchunhua@bupt.edu.cn

攻击分析与攻击检测

- 攻击检测原理
- 入侵检测技术
- 攻击检测实例

你怎么做检测？

- 假如你是一个新闻类门户网站的安全主管，你怎么检测？
  - DoS类
  - SQL注入、跨站等
  - 挂马、篡改类
  - 。。。
- 如果是一个公司内部服务，又有什么不同？

怎么做攻击检测？

- 为什么要检测？
- 是什么要检测？
- 凭什么来检测？
- 用什么来检测？
- 能检测出什么？

- **检测的意义**
- **检测的目的**
- **检测的依据**
- **检测的方法**
- **检测的结果**

检测的目的和意义

- 并不是所有威胁都要检测
  - 有些威胁无法检测，如搭线窃听
  - 有些威胁没必要检测，如搭线窃听
- 检测是为了发现和保护
  - 传统的各类攻击发现：DoS、扫描等
  - 非授权用户的威胁：伪造、伪装等
  - 基本功能的保证：量子密钥协商
  - 复杂攻击的分析：APT

检测的依据

- 攻击的参与者
  - 攻击者
  - 攻击手段
  - 攻击目标
  - 可能被利用的第三方
- 对应检测依据
  - 攻击者信息：IP、账号等
  - 攻击过程：网络数据、日志等
  - 攻击目标信息：IP、服务等
  - 可能被利用的第三方：关系

攻击分析与攻击检测——

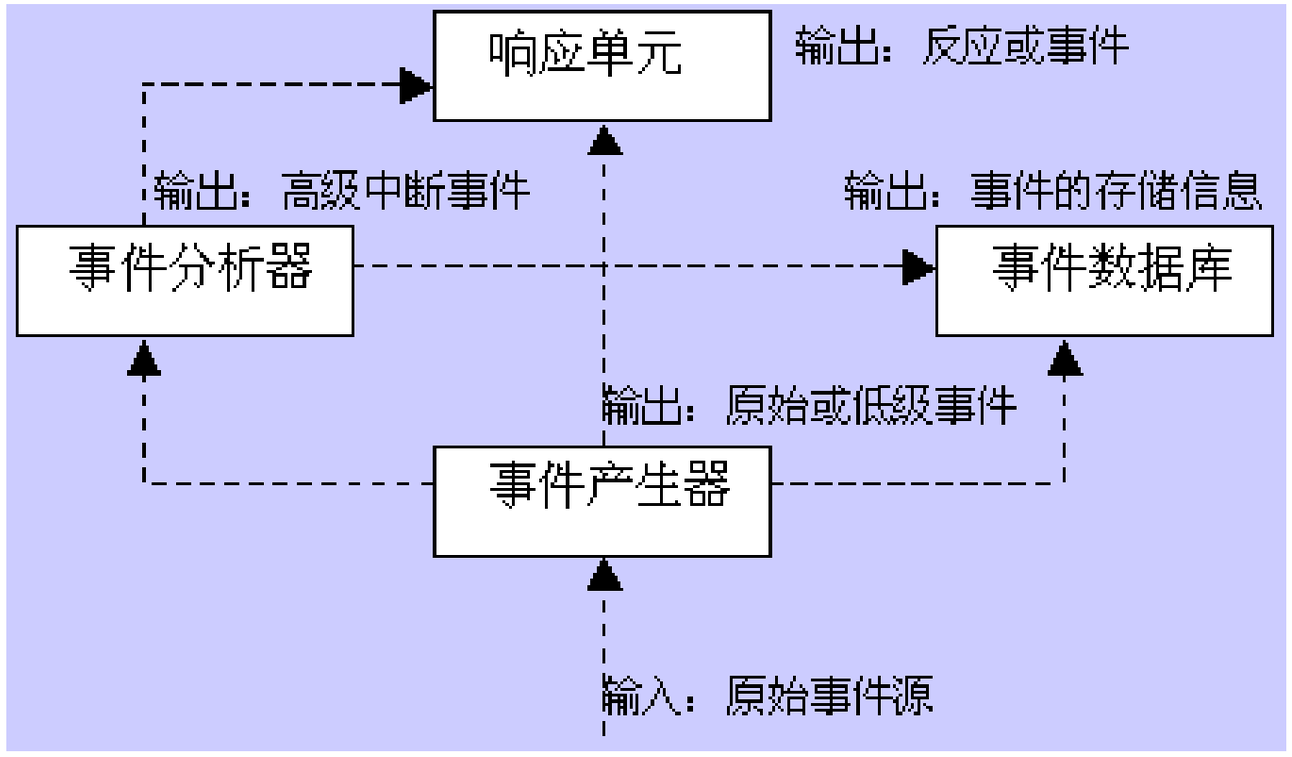
入侵检测结构

入侵检测结构

- CIDF

- Common Intrusion Detection Framework

- 由DARPA于1997年3月开始着手制定



- 事件产生器(Event Generators)

- 事件分析器(Event analyzers)

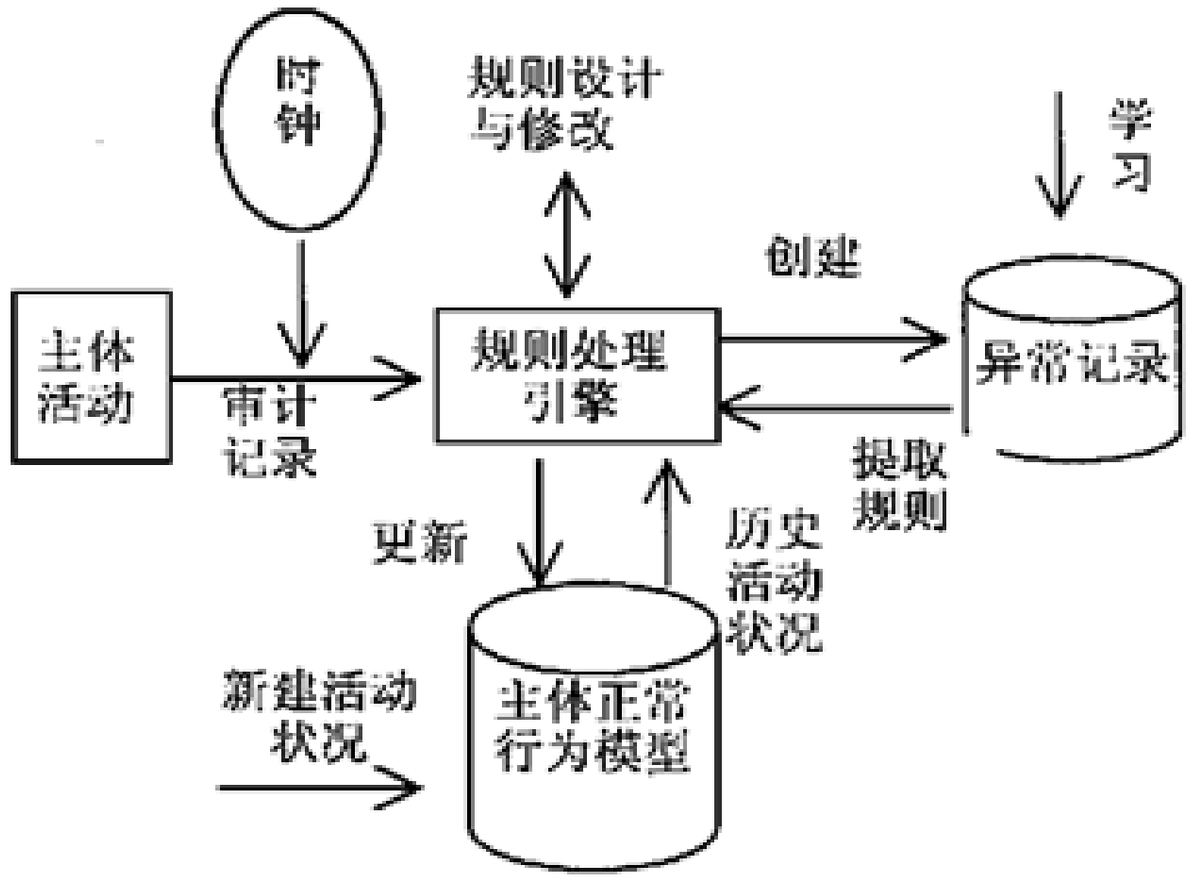
- 响应单元(Response units)

- 事件数据库(Event databases)

入侵检测结构

- Denning模型
- 1987年提出的一个通用入侵检测模型

- 主体(Subjects)：在目标系统上活动的实体，如用户。
- 对象(Objects)：指系统资源，如文件、设备、命令等。
- 审计记录(Audit records)：由主体、活动(主体对目标的操作)、异常条件(系统对主体的该活动的异常情况的报告)、资源使用状况(系统的资源消耗情况)和时间戳(Time-Stamp)等组成。
- 活动档案(Active Profile)：即系统正常行为模型，保存系统正常活动的有关信息。
- 异常记录(Anomaly Record)：由事件、时间戳和审计记录组成，表示异常事件的发生情况。
- 活动规则(Active Rule)：判断是否为入侵的准则及相应要采取的行动。。



IDWG

- Intrusion Detection Work Group
  - IDS系统之间、IDS和网管系统之间
  - 共享的数据格式
  - 统一的通信规程
- 草案
  - IDMEF(入侵检测消息交换格式)
  - IDXP（入侵检测交换协议）

攻击分析与攻击检测——

入侵检测技术

入侵检测系统

- 入侵检测是从计算机网络或计算机系统中的若干关键点搜集信息并对其进行分析，从中发现网络或系统中是否有违反安全策略的行为和遭到袭击的迹象的一种机制。

- IDS，Intrusion Detection System，入侵检测系统

入侵检测分类

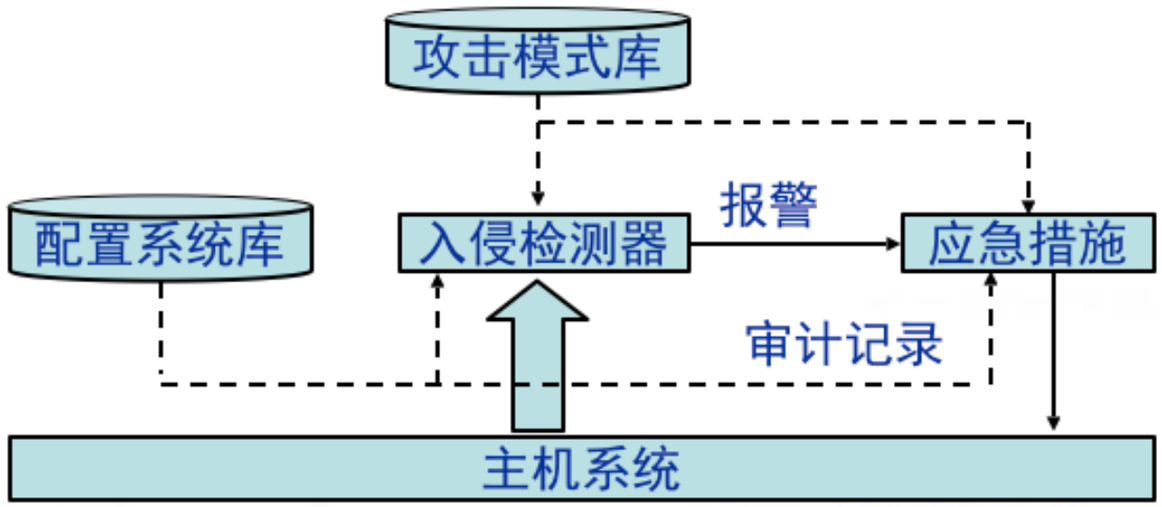
- 根据原始数据的来源
  - 基于主机的入侵检测系统：监控粒度更细、配置灵活、可用于加密的以及交换的环境
  - 基于网络的入侵检测系统：视野更宽 、隐蔽性好 、攻击者不易转移证据
- 根据检测原理
  - 异常入侵检测：根据异常行为和使用计算机资源的情况检测出来的入侵。
  - 误用入侵检测：利用已知系统和应用软件的弱点攻击模式来检测入侵。

入侵检测分类

- 根据体系结构
  - 集中式：多个分布于不同主机上的审计程序，一个入侵检测服务器
  - 等级式：定义了若干个分等级的监控区域，每个IDS负责一个区域，然后将当地的分析结果传送给上一级IDS
  - 协作式：将中央检测服务器的任务分配给多个基于主机的IDS，这些IDS不分等级，各司其职，负责监控当地主机的某些活动。
- 根据工作方式分类
  - 离线检测：非实时工作系统，在事件发生后分析审计事件，从中检查入侵事件。
  - 在线检测：对网络数据包或主机的审计事件进行实时分析，可以快速反应，保护系统的安全；但系统规模较大时，难以保证实时性。

主机IDS

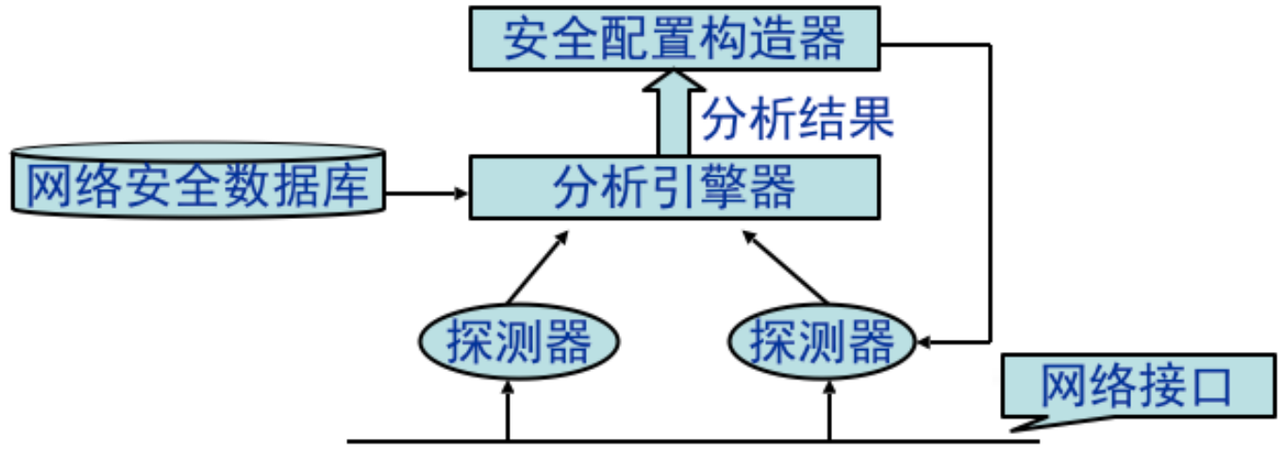
- HIDS：Host-Based IDS
- 检测的目标主要是主机系统和系统本地用户。检测原理是根据主机的审计数据和系统的日志发现可疑事件，检测系统可以运行在被检测的主机或单独的主机上。



- HIDS优点
  - 性能价格比高
  - 细腻性，审计内容全面
  - 适用于加密及交换环境
- HIDS缺点
  - IDS的运行影响服务器的性能
  - HIDS依赖性强，依赖于审计数据或系统日志准确性和完整性，以及安全事件的定义。

网络IDS

- NIDS：Network-Based IDS
- 根据网络流量、协议分析、单台或多台主机的审计数据检测入侵。



- NIDS缺点
  - 协同工作能力弱
  - 难以处理加密的会话
- NIDS优点：
  - 服务器平台独立性：监视通信流量而不影响服务器的平台的变化和更新；
  - 配置简单：只需要一个普通的网络访问接口即可；
  - 众多的攻击标识：探测器可以监视多种多样的攻击包括协议攻击和特定环境的攻击。

入侵检测技术

异常检测技术　　误用检测技术

异常检测技术

- 思想：任何正常人的行为有一定的规律，而入侵会引起用户或系统行为的异常
- 需要考虑的问题：
  - 选择哪些数据来表现用户的行为
  - 通过以上数据如何有效地表示用户的行为，主要在于学习和检测方法的不同
  - 考虑学习过程的时间长短、用户行为的时效性等问题
- 典型算法：统计分析（均值、偏差、Markov过程）

异常检测技术

- 优点
  - 可以检测到未知的入侵
  - 可以检测冒用他人帐号的行为
  - 具有自适应，自学习功能
  - 不需要系统先验知识
- 缺点
  - 漏报、误报率高：入侵者可以逐渐改变自己的行为模式来逃避检测；合法用户正常行为的突然改变也会造成误警
  - 统计算法的计算量庞大，效率很低
  - 统计点的选取和参考库的建立比较困难

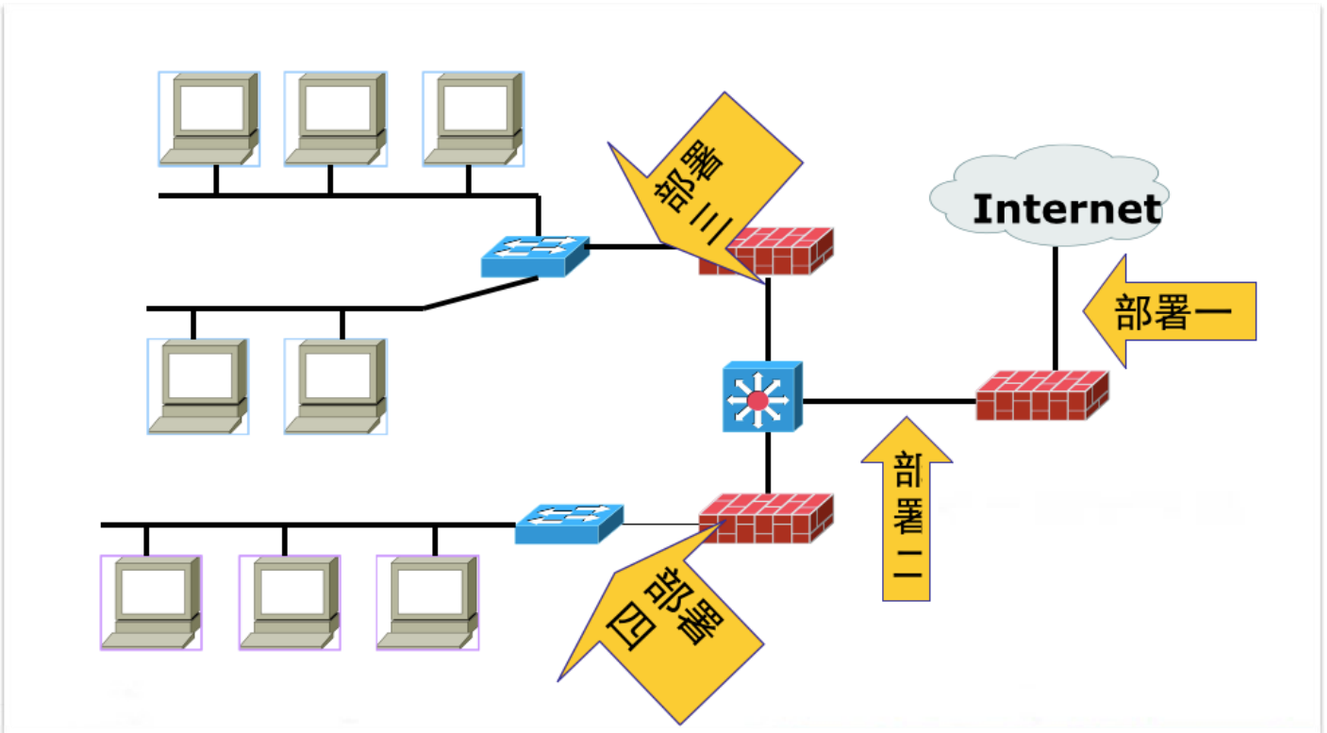
误用检测技术

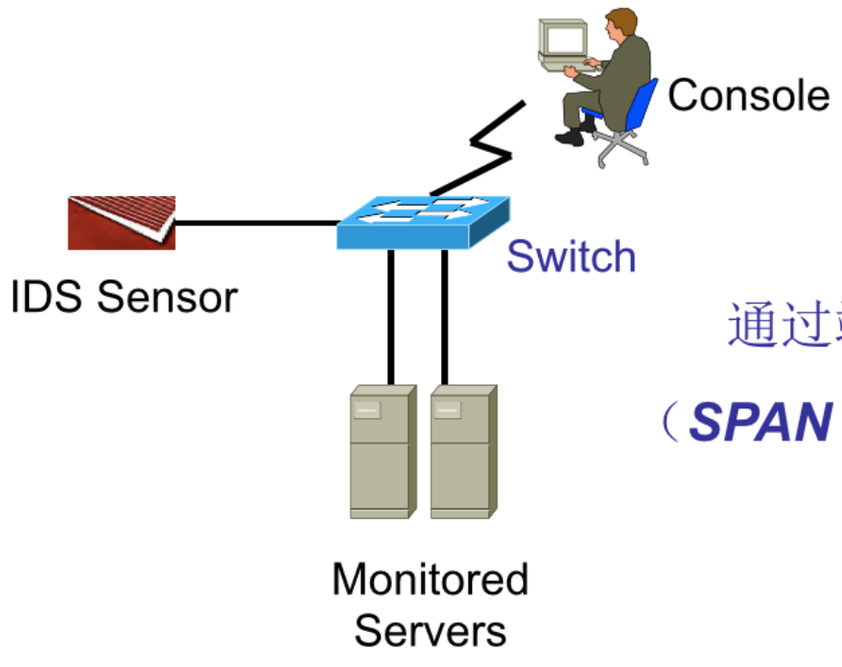
- 思想：主要是通过某种方式预先定义入侵行为，然后监视系统，从中找出符合预先定义规则的入侵行为
- 误用信号需要对入侵的特征、环境、次序以及完成入侵的事件相互间的关系进行描述
- 重要问题
  - 如何全面的描述攻击的特征
  - 如何排除干扰，减小误报
  - 解决问题的方式
- 典型算法：专家系统、模型推理、完整性分析

误用检测技术

- 优点：
  - 算法简单
  - 系统开销小
  - 准确率高
  - 效率高
- 缺点：
  - 被动：只能检测出已知攻击 、新类型的攻击会对系统造成很大的威
  - 模式库的建立和维护难：模式库要不断更新，知识依赖于硬件平台、操作系统和系统中运行的应用程序等

典型部署





通过端口镜像实现

（***SPAN / Port Monitor***）

攻击分析与攻击检测——

攻击分析实例

检测方法实例一：snort

- 典型的误用检测技术
- 需要准确理解攻击模式并撰写检测规则
- 无法应对未知攻击，造成漏报
- 检测效率高、准确率高
- 攻击规则更新要求高，好在攻击方式变化不剧烈

方法：基于攻击特征匹配的原理，准确度高，误报率低，无法检测未知攻击，典型的误用检测技术。

检测方法实例一：snort

EXAMPLE

Rule Header	`alert tcp $EXTERNAL_NET $HTTP_PORTS -> $HOME_NET any`
Message	`msg: "BROWSER-IE Microsoft Internet Explorer CacheSize exploit attempt";`
Flow	`flow: to_client,established;`
Detection	`file_data; content:"recordset"; offset:14; depth:9; content:".CacheSize"; distance:0; within:100; pcre:"/CacheSize\s*=\s*/"; byte_test:10,>,0x3ffffffe,0,relative,string;`
Metadata	`policy max-detect-ips drop, service http;`
References	`reference:cve,2016-8077;`
Classification	`classtype: attempted-user;`
Signature ID	`sid:65535;rev:1;`

检测方法实例一：snort

DoS检测——Land

- 攻击目的：land 攻击是一种使用相同的源和目的主机发送数据包到某台机器的攻击，结果通常使存在漏洞的机器崩溃。
- 攻击原理：一个特别打造的SYN包中的源地址和目标地址都被设置成某一个服务器地址，这时将导致接受服务器向它自己的地址发送SYN一ACK消息，结果这个地址又发回ACK消息并创建一个空连接，每一个这样的连接都将保留直到超时掉。
- 检测方法：**判断网络数据包的源地址和目标地址是否相同**。

注：实例规则为示意性规则，sid非snort官方编号。

Snort规则实例一

```
alert ip $HOME_NET any -> $HOME_NET any
(msg:“DoS Land attack”; flags:S; flow: stateless
classtype:attempted-dos; sid:6001; rev:1;sameip)
```

- 过滤筛选ip数据包，来自本地地址的任意端口，发往本地地址的任意端口，在报警和包日志中打印“DoS Land attack”；数据包中flags字段的值是SYN。
- 规则类别标识是attempted-dos；snort规则id号为6001；rev是用来识别规则修改的次数，为1。
- sameip关键字允许规则检测源IP和目的IP是否相等。

Snort规则实例二

扫描检测——TCP NULL扫描

- 攻击目的：可以判断目标主机操作系统是windows还是类Unix系统。
- 攻击原理：根据RFC 793，类Unix系统接收到没有设置任何标志位的数据包，端口关闭的情况下舍弃掉该数据包并且发送一个RST数据包，端口开放的话不会响应；而Windows系统不满足RFC 793，当收到该数据包的情况下，无论端口开放或者关闭，都会响应一个RST数据包给发送方。
- 检测方法：验证数据包中所有标志位是否都为0。

Snort规则实例二

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (ms
g:“SCAN NULL” ;flags:0; flow: from_client; clas
stype:attempted-recon; sid:6002; rev:1;)
```

- 过滤筛选tcp协议的数据包，来自外部网络的任意端口，发往本地的任意端口，并且**tcp包标志位全部为0**，在报警和包日志中打印“SCAN NULL”
- Flow表示规则只应用到来自客户端的TCP数据包；规则类别标识是attempted-recon（Attempted Information Leak）；
- 规则id为6002，规则修改次数为1。

检测方法实例一：snort

漏洞利用检测——SQL注入漏洞

- 攻击目的：恶意用户通过构造特殊字符串从而从服务器获得敏感信息。

- 攻击原理：#可以注释掉SQL语句后面的一行SQL代码，相当于去掉了后面的where条件，而且前面的1=1永远都是成立的，即where子句总是为真，例如select * from users where username='' or 1=1 #' and password=md5('')

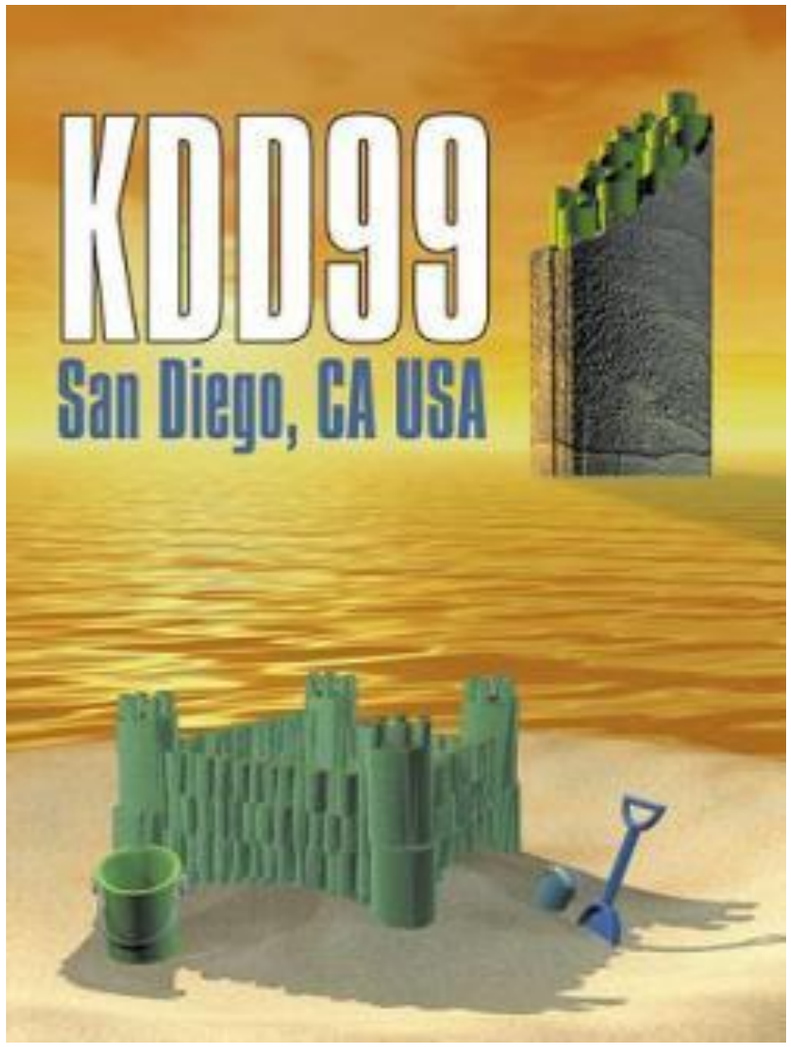
- 检测方法：验证数据包中的内容中是否存在“' or 1=1 #”

Snort规则实例三

alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:“SQL Injection found”; flow:from_client, established; content:“’ %20and%201=1#”; classtype: web-application-attack; sid:6003; rev:1;)

- 过滤筛选来自客户端的tcp数据包，来自外部网络的任意端口，发往本地的任意端口，并且包的净荷中含有“’ %20and%201=1#”，在报警和包日志中打印“SQL Injection found”
- Flow表示规则只应用到已经建立的来自客户端的TCP连接；
- 规则类别标识是web-application-attack；规则id为6003，规则修改次数为1

检测方法实例二：KDD Cup



- Normal users have rules
- Attacks have rules
- Rules hide in the hosts and network data.
- ML can help us to find these rules from the training data set.

方法：机器学习的方法，需要大量先验数据，样本要全面，对针对性的攻击检测难度大。

Why we use Meachine Learning?

- This is a typical prediction model of machine learning.

- common techniques:
  - Neural network,decision tree
  - SVM, KNN,clustering
  - ……　　……

检测方法实例二：KDD Cup

KDD Cup is the annual Data Mining and Knowledge Discovery competition organized by ACM Special Interest Group on **Knowledge Discovery and Data Mining**, the leading professional organization of data miners. Year to year archives including datasets, instructions, and winners are available for most years.

检测方法实例二：KDD Cup

- KDD-Cup 2008, Breast cancer
- KDD-Cup 2007, Consumer recommendations
- KDD-Cup 2006, Pulmonary embolisms detection from image data
- KDD-Cup 2005, Internet user search query categorization
- KDD-Cup 2004, Particle physics; plus Protein homology prediction
- KDD-Cup 2003, Network mining and usage log analysis
- KDD-Cup 2002, BioMed document; plus Gene role classification
- KDD-Cup 2001, Molecular bioactivity; plus Protein locale prediction.
- KDD-Cup 2000, Online retailer website clickstream analysis
- **KDD-Cup 1999, Computer network intrusion detection**
- KDD-Cup 1998, Direct marketing for profit optimization
- KDD-Cup 1997, Direct marketing for lift curve optimization

KDD Cup 1999

Introduction

KDD Cup 1999: Computer network intrusion detection

The task for the classifier learning contest organized in conjunction with the KDD'99 conference was to learn a predictive model (i.e. a classifier) capable of distinguishing between legitimate and illegitimate connections in a computer network.

KDD Cup 1999 Data

- The 1998 DARPA Intrusion Detection Evaluation Program was prepared and managed by MIT Lincoln Labs.

- **The objective was to survey and evaluate research in intrusion detection.**

- A standard set of data to be audited, which includes a wide variety of intrusions simulated in a military network environment, was provided.

- The 1999 KDD intrusion detection contest uses a version of this dataset.

KDD Cup 1999 Data

- **Lincoln Labs set up an environment to acquire nine weeks of raw TCP dump data for a local-area network (LAN) simulating a typical U.S. Air Force LAN.** They operated the LAN as if it were a true Air Force environment, but peppered it with multiple attacks.

- The raw **training data** was about four gigabytes of compressed binary TCP dump data from **seven weeks** of network traffic. This was processed into **about five million connection records**. Similarly, the **two weeks of test data** yielded around **two million connection records**.

KDD Cup 1999 Data Data Set

- This database contains a standard set of data to be audited, which includes a wide variety of intrusions simulated in a military network environment.

Data Set Characteristics:	Multivariate	Number of Instances:	4000000	Area:	Computer
Attribute Characteristics:	Categorical, Integer	Number of Attributes:	42	Date Donated	1999-01-01
Associated Tasks:	Classification	Missing Values?	N/A	Number of Web Hits:	64262

Remark:42 Number of Attributes = 41 kinds of Features + 1 kind of results

http://www.ll.mit.edu/ideval/data/1999data.html
http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

KDD Cup 1999 Data

- The datasets contain a total of 24 training attack types, with an additional 14 types in the test data only.

- Attacks fall into four main categories:
  - **probing:** surveillance and other probing, e.g., port scanning.
  - **DOS:** denial-of-service, e.g. syn flood;
  - **U2R:** unauthorized access to local superuser (root) privileges, e.g., various ``buffer overflow'' attacks;
  - **R2L:** unauthorized access from a remote machine, e.g. guessing password;

KDD Cup 1999 Data Data Set

PROBE	ipsweep
	mscan
	nmap
	portsweep
	saint
	satan

DOS	apache2
	back
	land
	mailbomb
	neptune
	pod
	processtable
	smurf
	teardrop
	udpstorm

KDD Cup 1999 Data Data Set

U2R	buffer_overflow
	httptunnel
	loadmodule
	perl
	ps
	rootkit
	sqlattack
	xterm

R2L	ftp_write
	guess_passwd
	imap
	multihop
	named
	phf
	sendmail
	snmpgetattack
	snmpguess
	spy
	warezclient
	warezmaster
	worm
	xlock
	xsnoop

TABLE 1: Basic features of individual TCP connections

Basic features of individual TCP Connections			
ID	Feature Name	Description	Type
1	duration	length (number of seconds) of the connection	continuous
2	protocol_type	type of the protocol, e.g. tcp, udp, etc.	discrete
3	service	network service on the destination, e.g., http, telnet, etc.	discrete
4	src_bytes	number of data bytes from source to destination	continuous
5	dst_bytes	number of data bytes from destination to source	continuous
6	flag	normal or error status of the connection	discrete
7	land	1 if connection is from/to the same host/port; 0 otherwise	discrete
8	wrong_fragment	number of ``wrong'' fragments	continuous
9	urgent	number of urgent packets	continuous

TABLE 2: Content features within a connection suggested by domain knowledge

Content Features Within a Connection Suggested by Domain Knowledge			
ID	Feature Name	Description	Type
10	hot	number of ``hot'' indicators	continuous
11	num_failed_logins	number of failed login attempts	continuous
12	logged_in	1 if successfully logged in; 0 otherwise	discrete
13	num_compromised	number of ``compromised'' conditions	continuous
14	root_shell	1 if root shell is obtained; 0 otherwise	discrete
15	su_attempted	1 if ``su root'' command attempted; 0 otherwise	discrete
16	num_root	number of ``root'' accesses	continuous
17	num_file_creations	number of file creation operations	continuous
18	num_shells	number of shell prompts	continuous
19	num_access_files	number of operations on access control files	continuous
20	num_outbound_cmds	number of outbound commands in an ftp session	continuous
21	is_hot_login	1 if the login belongs to the ``hot'' list; 0 otherwise	discrete
22	is_guest_login	1 if the login is a ``guest''login; 0 otherwise	discrete

TABLE 3: Traffic features computed using a two-second time window

Traffic Features Computed Using a Two-second Time Window			
ID	Feature Name	Description	Type
23	count	number of connections to the same host as the current connection in the past two seconds Note: The following features refer to these same-host connections.	continuous
24	serror_rate	% of connections that have ``SYN'' errors	continuous
25	rerror_rate	% of connections that have ``REJ'' errors	continuous
26	same_srv_rate	% of connections to the same service	continuous
27	diff_srv_rate	% of connections to different services	continuous
28	srv_count	number of connections to the same service as the current connection in the past two seconds Note: The following features refer to these same-service connections.	continuous
29	srv_serror_rate	% of connections that have ``SYN'' errors	continuous
30	srv_rerror_rate	% of connections that have ``REJ'' errors	continuous
31	srv_diff_host_rate	% of connections to different hosts	continuous

TABLE 4: Host Features Based on the Purpose of the TCP Connection

Host Features Based on the Purpose of the TCP Connection			
ID	Feature Name	Description	Type
32	dst_host_count	[0, 255]	continuous
33	dst_host_srv_count	[0, 255]	continuous
34	dst_host_same_srv_rate	[0.00, 1.00]	continuous
35	dst_host_diff_srv_rate	[0.00, 1.00]	continuous
36	dst_host_same_src_port_rate	[0.00, 1.00]	continuous
37	dst_host_srv_diff_host_rate	[0.00, 1.00]	continuous
38	dst_host_serror_rate	[0.00, 1.00]	continuous
39	dst_host_srv_serror_rate	[0.00, 1.00]	continuous
40	dst_host_rerror_rate	[0.00, 1.00]	continuous
41	dst_host_srv_rerror_rate	[0.00, 1.00]	continuous

Analysis of kdd99 database

- **Learning base : 4 connections have the same 41 attributes but the label is different**

- **0,icmp,ecr_i,SF,8,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,1,1,1.00,0.00,1.00,0.00,0.00,0.00,0.00,0.00,ipsweep.148774**

- **0,icmp,ecr_i,SF,8,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,1,1,1.00,0.00,1.00,0.00,0.00,0.00,0.00,0.00,portsweep.345836**

- **0,icmp,tim_i,SF,564,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,2,2,1.00,0.00,1.00,0.00,0.00,0.00,0.00,0.00,normal.143855**

- **0,icmp,tim_i,SF,564,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,2,2,1.00,0.00,1.00,0.00,0.00,0.00,0.00,0.00,pod.345952**

Analysis of kdd99 database

- Test base (corrected file)
- 71 distinct connections have the same attributes but have the different labels.
- 71503 (22.99% of the total) connections have the same attributes but appear the different labels
- 3 ipsweep (Probing) attack connections have the same attributes of those of smurf (DoS) attack (56608 connections)
  - 3 (0.07%) Probing attacks cannot be detected (classifed as DoS attack instead)

Analysis of kdd99 database

- Test base (corrected file) :
  - 7563 (97.7% of the total) connections of the snmpgetattack attack have the same attributes of those of normal
  - 2.3% of the snmpgetattack have similar attributes as normal, (but not all the same)
  - 7563 (46.72% of the total) R2L attack cannot be detected (they are classifed as normal)

Intrusion detection and Identification based on KDD99 data

- Building the normal model based on normal data for intrusion detection

- Building individual attack model based on corresponding attack data for intrusion identification

EXAMPLE:

KNN BASED INTRUSION DETECTION

K-NEAREST NEIGHBOR CLASSIFICATION (KNN)

- Unlike all the previous learning methods, kNN does not build model from the training data.

- To classify a test instance d, define k-neighborhood P as k nearest neighbors of d

- Count number n of training instances in P that belong to class c_j

- Estimate $\Pr(c_j|d)$ as n/k

- No training is needed. Classification time is linear in training set size for each test case.

BASIC K-NEAREST NEIGHBOR CLASSIFICATION

- Training method:
  - Save the training examples
- At prediction time:
  - Find the k training examples $(x_1, y_1), \cdots (x_k, y_k)$ that are closest to the test example x
  - Predict the most frequent class among those y_i's.
- Improvements:
  - *Weighting* examples from the neighborhood
  - Measuring "*closeness*"
  - Finding "close" examples in a large training set *quickly*

KNN ALGORITHM

Algorithm kNN(D, d, k)

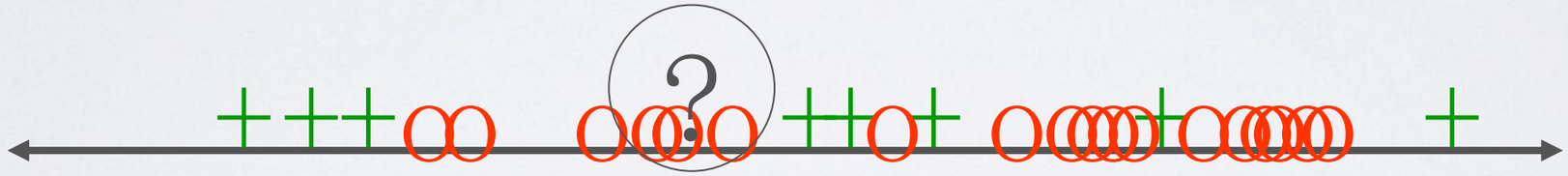
1 Compute the distance between d and every example in D;

2 Choose the k examples in D that are nearest to d, denote the set by P ($\subseteq D$);

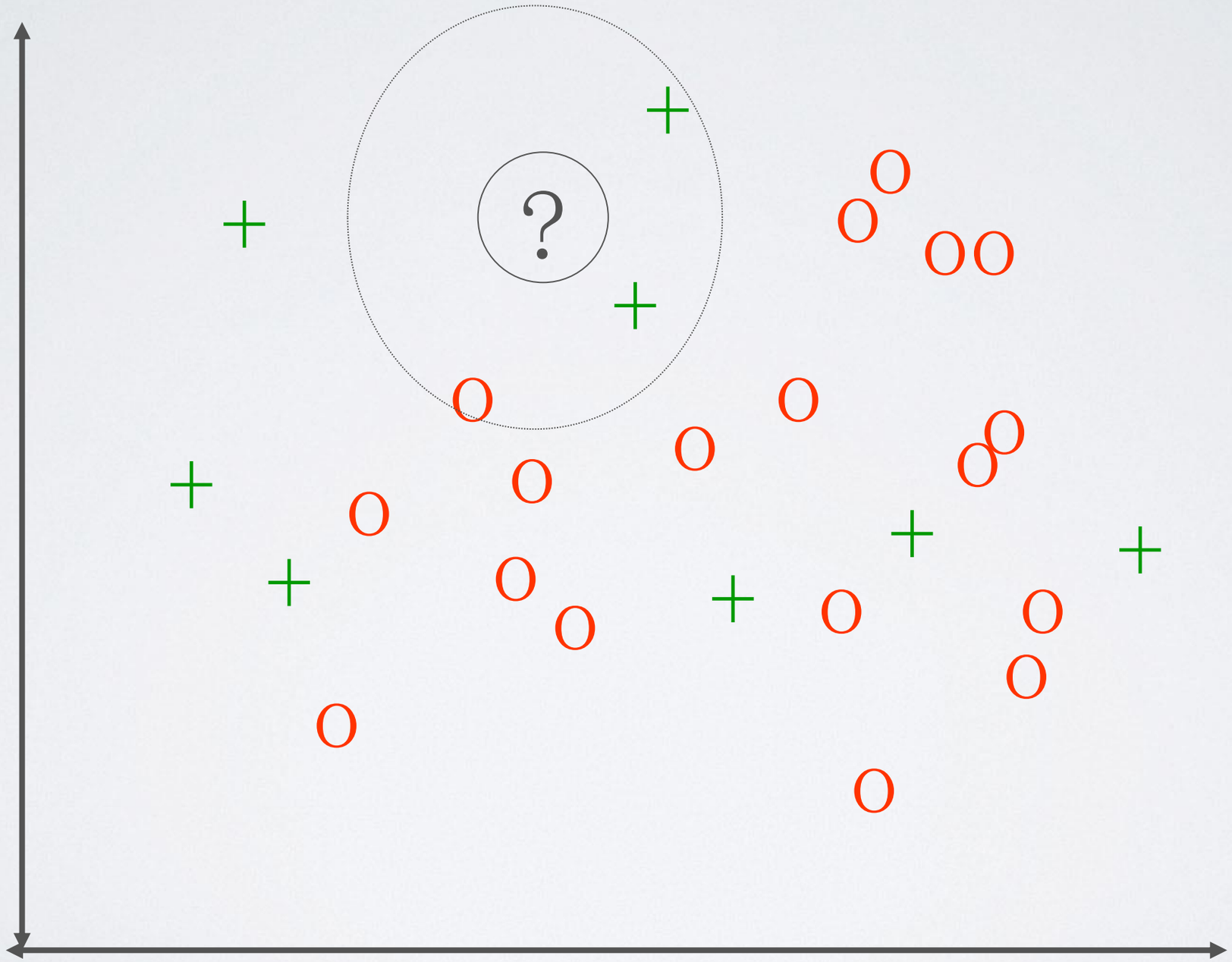
3 Assign d the class that is the most frequent class in P (or the majority class);

- k is usually chosen empirically via a validation set or cross-validation by trying a range of k values.
- Distance function is crucial, but depends on applications.

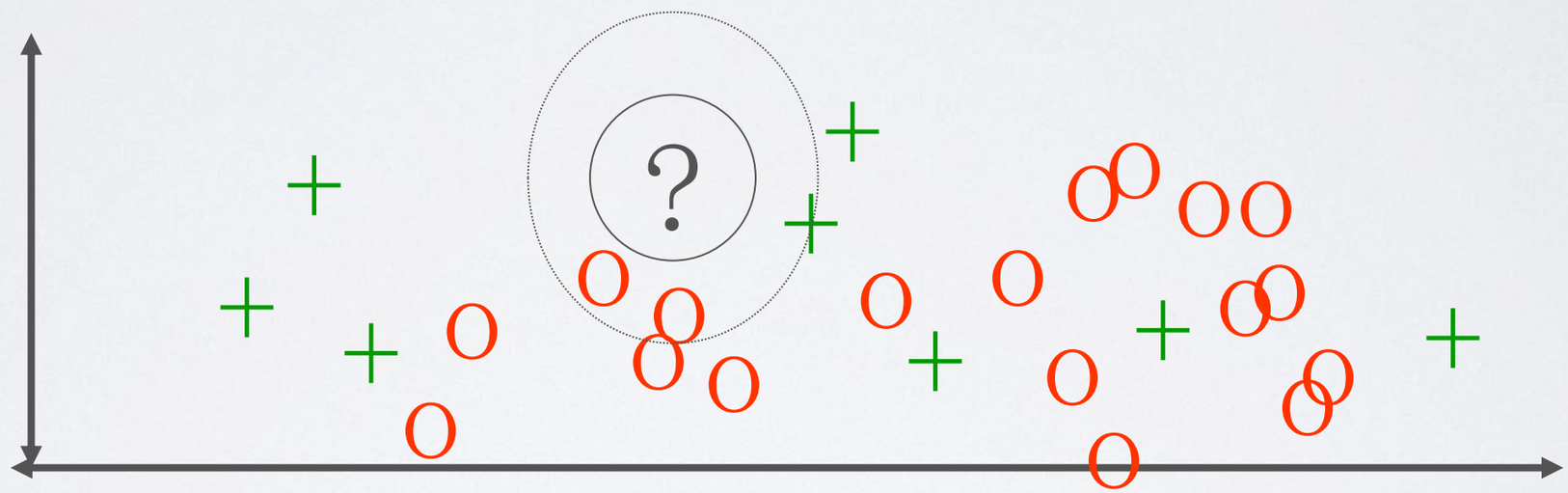
K-NN AND IRRELEVANT FEATURES



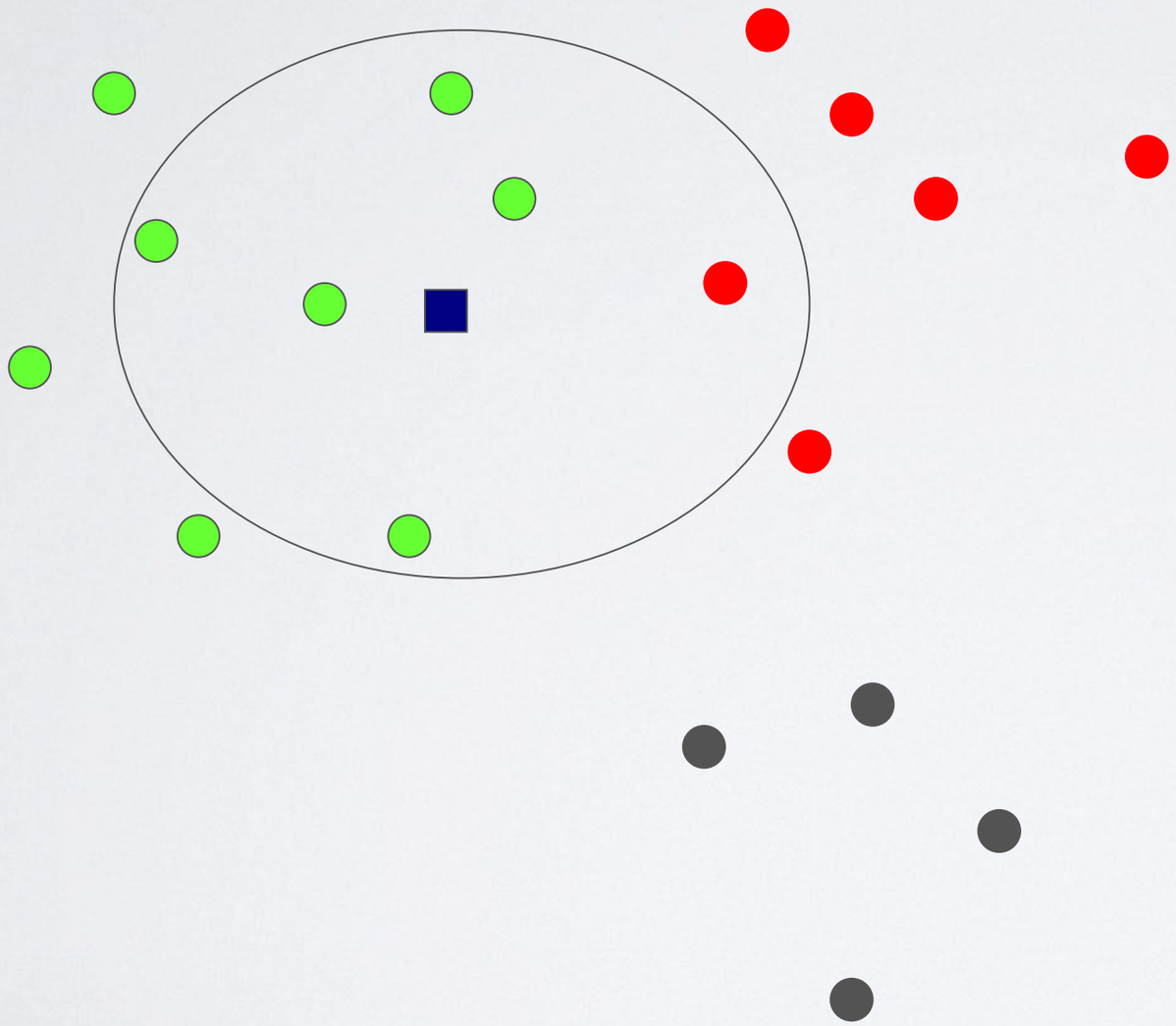
K-NN AND IRRELEVANT FEATURES



K-NN AND IRRELEVANT FEATURES



EXAMPLE: K=6 (6NN)



- Government
- Science
- Arts

A new point■
Pr(science| ■)?

DISCUSSIONS

- kNN can deal with complex and arbitrary decision boundaries.

- Despite its simplicity, researchers have shown that the classification accuracy of kNN can be quite strong and in many cases as accurate as those elaborated methods.

- kNN is slow at the classification time

- kNN does not produce an understandable model

KNN BASED INTRUSION DETECTION

- **Building normal behavioral model**
  - Calculate the distances between each test vector t and each vector in the training data set by using Euclidean distance:

$$dis_{eu}(\mathbf{t},\mathbf{x}_j) = \left\|\mathbf{t}-\mathbf{x}_j\right\| = \sqrt{\sum_{i=1}^{M}(t_i - x_{ij})^2}$$

  - Sort the distance and choose the *k* nearest neighbors.
  - Average the *k* closest distance scores as the *anomaly index*.
- **Detection**
  - If the *anomaly index* of a test sequence vector **t** is above a threshold **ε**
    - the test sequence is then classified as abnormal.
    - otherwise it is considered as normal.

KNN BASED INTRUSION IDENTIFICATION

Define normal and individual attack data sets as $D_1, D_2, \cdots, D_l$;

Identification:

For each test vector **t do**

1. 计算样本与训练集各样本的距离

Calculate $dis_{eu}(\mathbf{t}, \mathbf{x}_j)$ for $\mathbf{x}_j$ in each training set;

2. 找到最近的k个邻居

Find k smallest scores of $dis_{eu}(\mathbf{t}, \mathbf{x}_j)$ as k-nearest neighbors;

If more than a half of k nearest neighbors correspond to a specific attack type A_k **then**

3. 如果超过k/2个属于Ak类，则认为属于Ak

t is identified as A_k

Else If the number of smallest distance that corresponds to an attack type A_p is greater than those of others **then**

4. 否则选最大的Ap作为类别

t is identified as A_p

Else then

t is identified as a new attack

5. 都不满足就是新类型

End If

End For

KNN Based intrusion detection

Attack type	Attack category	Identification Rate (%)		
		*k*NN		
		k=5	*k*=7	*k*=9
guess_ passwd	R2L	92.3	92.3	92.3
warezclient	R2L	100	100	100
warezmaster	R2L	80	80	80
back	DoS	98.5	99.5	98
neptune	DoS	99.8	99.8	97.7
pod	DoS	100	96.9	100
smurf	DoS	100	100	100
teardrop	DoS	97.7	99.4	97.8
buffer overflow	U2R	80	80	80
ipsweep	Probe	97.6	99.2	97.6
nmap	Probe	12.9	12.9	12.9
portsweep	Probe	100	100	100
satan	Probe	88.2	88.2	88.2

检测方法实例三：恶意APP

- 恶意APP检测
  - 检测原理
  - 检测方法
  - 结果（图+解释）

检测方法实例三：恶意APP

静态检测方法

- APP反编译，解析代码文件，提取应用特征；
- 采用机器学习算法建立智能检测模型；
- 简单，效率高，但无法解析加固的恶意应用。

应用特征

- 应用属性
- 应用签名
- 应用权限
- 应用组件
- 行为API

APK 反编译

源代码文件
AndroidManifest.xml
.java

提取特征

检测方法实例三：恶意APP

动态检测方法

- 应用运行，监控运行状态，提取行为特征；
- 采用机器学习算法建立智能检测模型；
- 可检测加固应用，但效率低，消耗系统资源多，难以覆盖全部执行路径；
- 与静态检测方式互补。

应用特征

- 行为序列
- 系统广播
- 进程通信
- 网络流量
- 系统资源调用

提取特征

应用运行

检测方法实例三：恶意APP

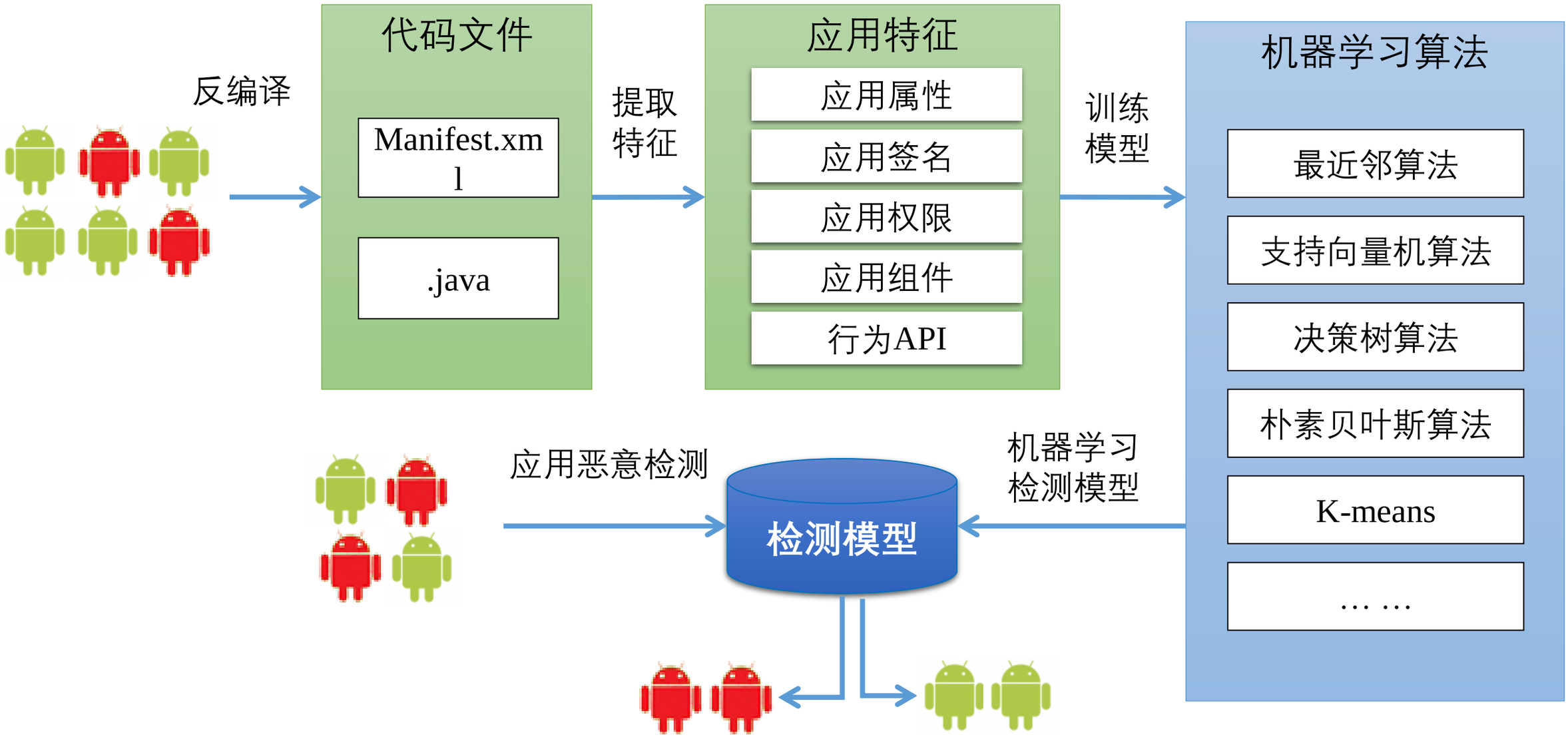
静态检测工具

- AXMLPrinter2工具解析AndroidManifest.xml文件;
- dex2jar工具将DEX反编译成.java源码，jd-gui工具查看java源码;
- 集成工具：APKtool、ApkToolkit、ApkAnalyser、Dexdump、Dedexer、androguard、apkinspector等。

动态检测工具

- 检测环境：沙箱Sandbox;
- 集成工具：IDA PRO、ZjDroid、Inspeckage、DynamicAPK、apktool + eclipse 等。

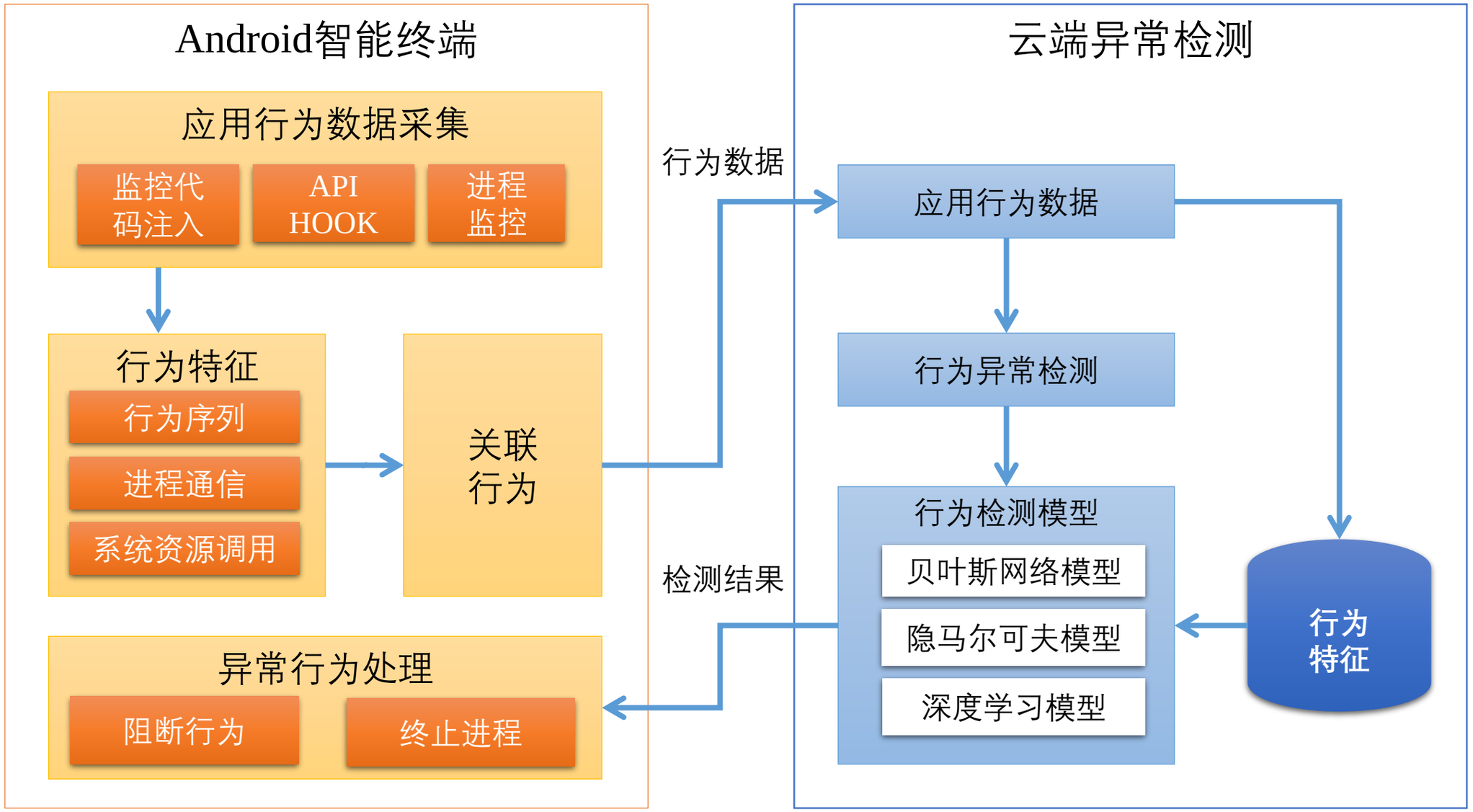
检测方法实例三：恶意APP



方法：大量标记样本，分析并提取应用特征，通过机器学习的方法寻找不同类别样本的分类界面。

检测方法实例三：恶意APP

Android恶意应用动态检测模型



检测方法实例三：恶意APP

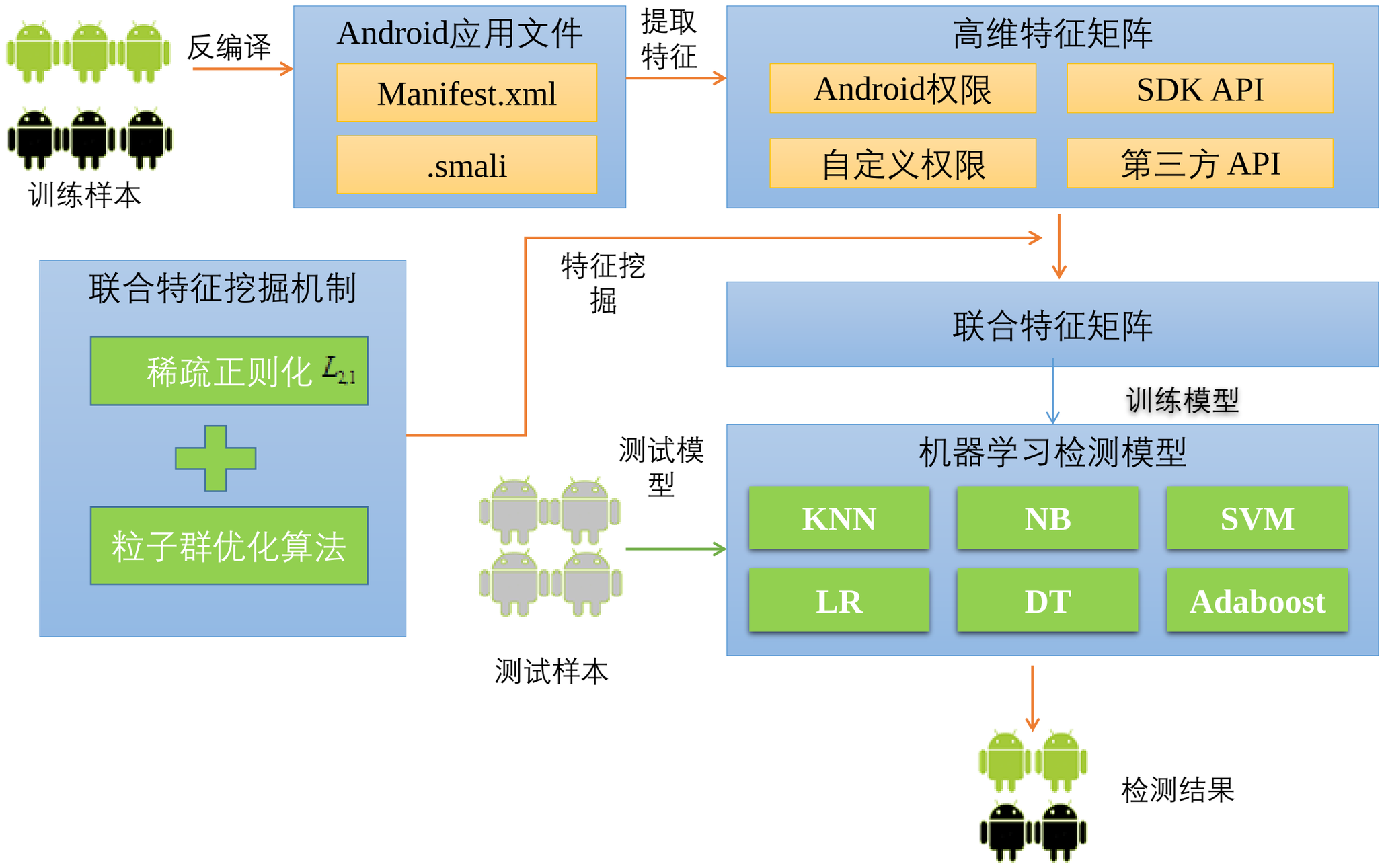
基于联合特征挖掘的Android恶意应用检测技术

检测原理：

- 1. 为了获得更多的分类信息，对训练集中的样本**提取更细粒度的特征**，这导致特征维数较大，且包含大量冗余特征。
- 2. 采用**特征选择算法**，从特征全集中选择包含最多分类信息的组合特征子集。
- 3.基于**机器学习分类算法**，对测试集中的Android恶意应用进行识别。

检测方法实例三：恶意APP

基于联合特征挖掘的Android恶意应用检测技术



检测方法实例三：恶意APP

稀疏正则化算法：

样本集合A 样本标签矩阵 Y

求解目标： $\min_{U} \| U \|_{2,1}$ $s.t.\ A^T U = Y$ $\| U \|_{2,1} = \sum_{i=1}^{d} \sqrt{\sum_{j=1}^{w} u_{ij}^2}$

拉格朗日算子求解： $L(U) = \| U \|_{2,1} - Tr(\Lambda^T (AU - Y))$

$$\frac{\partial L(U)}{\partial (U)} = 2DU - A^T \Lambda = 0 \qquad d_{ii} = \frac{1}{2 \| u^i \|_2}$$

$$2AU - AD^{-1}A^T \Lambda = 0$$

$$2Y - AD^{-1}A^T \Lambda = 0$$

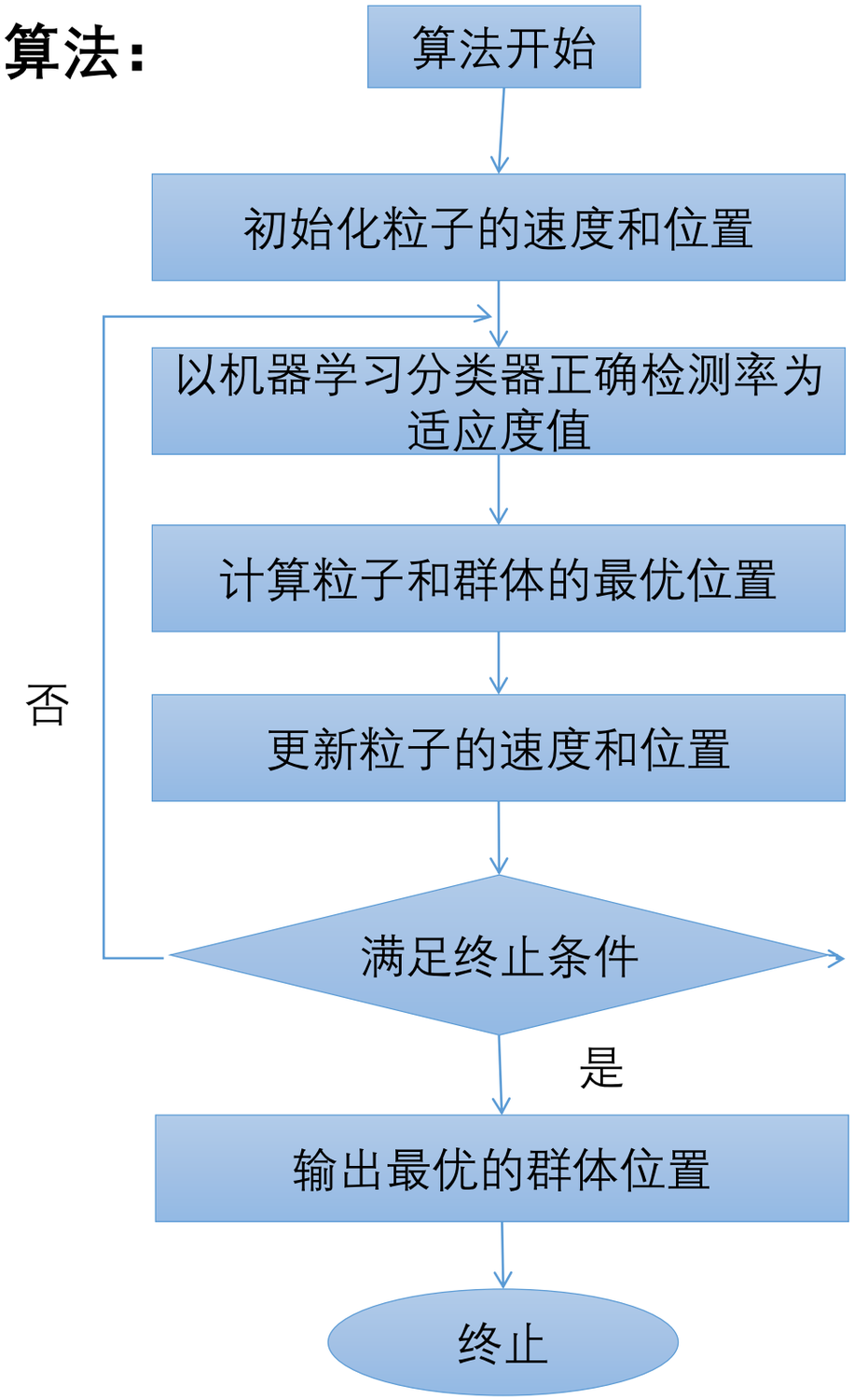
$$\Lambda = 2(AD^{-1}A^T)^{-1} Y$$

$$U = D^{-1}A^T (AD^{-1}A^T)^{-1} Y$$

注：使用稀疏正则化算法进行特征提取。降维，可以大致分为特征选择(feature selection)和特征提取(feature extraction)两大类。

检测方法实例三：恶意APP

粒子群优化算法：



速度和位置更新公式：

$$v_{ij} = w \cdot v_{ij} + c_1 \cdot rand() \cdot (p_{ij} - x_{ij}) + c_2 \cdot rand() \cdot (p_{gj} - x_{ij})$$

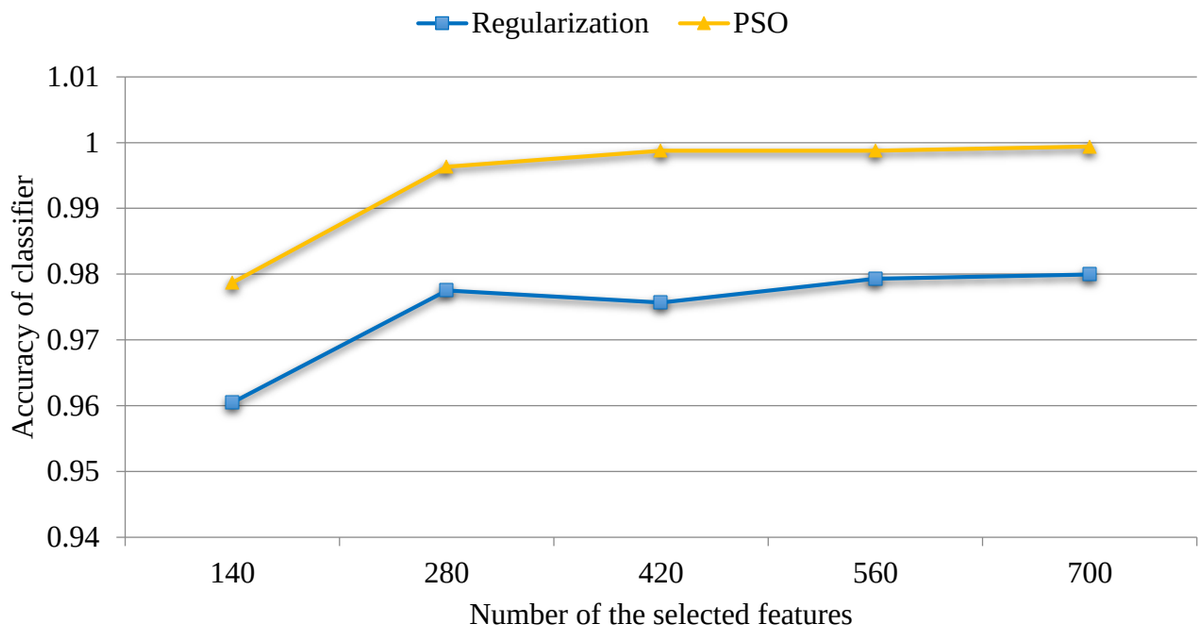
$$x_{ij} = x_{ij} + v_{ij}$$

检测方法实例三：恶意APP

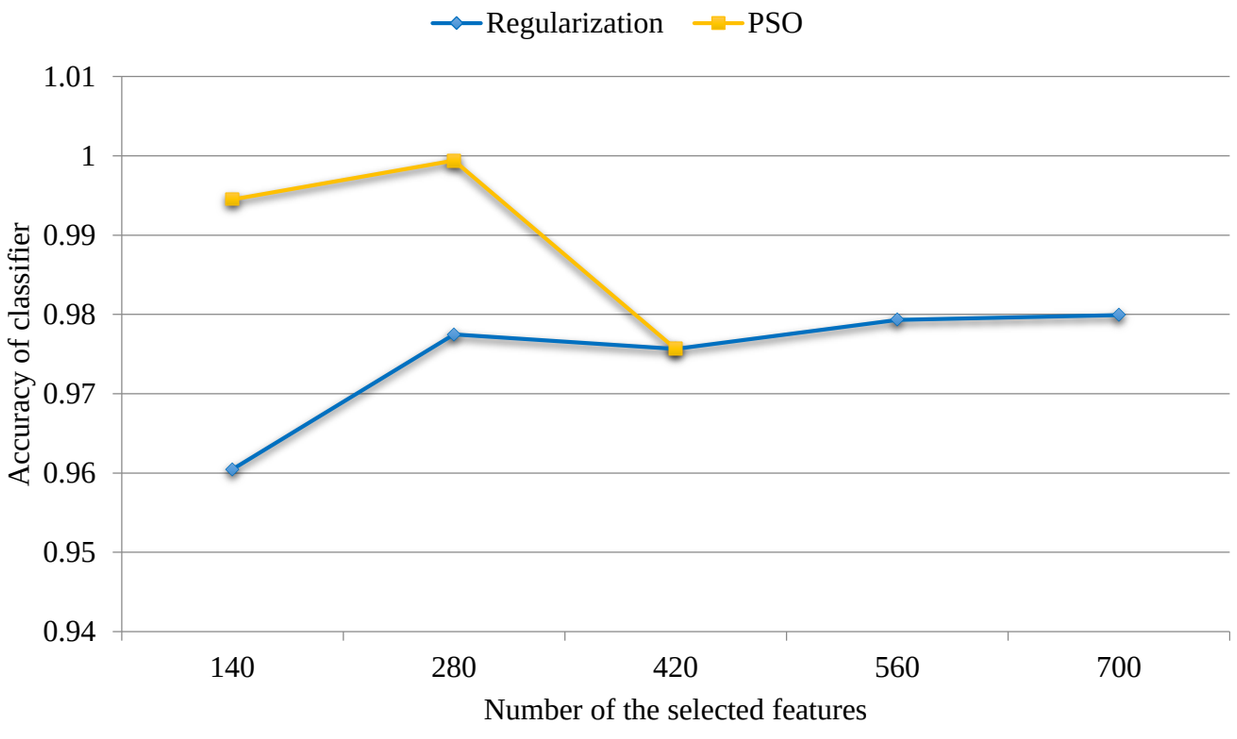
正则化算法和粒子群优化算法挖掘出的联合特征top20，不同算法挖掘的特征组合不同。

Regularization method	PSO method
Landroid/telephony/SmsManager;->getDefault	Landroid/text/SpannableStringBuilder;->append
android.permission.SEND_SMS	Ljava/util/Queue;->clear
android.permission.SYSTEM_ALERT_WINDOW	Ljava/lang/Thread;->toString
Landroid/telephony/SmsManager;->divideMessage	Ljava/lang/reflect/Field;->getAnnotation
Landroid/content/Intent;->setDataAndType	Landroid/os/ParcelFileDescriptor;->writeToParcel
Landroid/content/Context;->getApplicationInfo	Landroid/location/Location;->getAccuracy
Ljava/io/FileNotFoundException;->toString	Ljava/net/URL;->openStream
Ljava/util/LinkedHashSet;->add	Landroid/database/DataSetObserver;->onChanged
Landroid/content/IntentFilter;->setPriority	Landroid/os/Parcel;->writeParcelable
Ljava/lang/Integer;->valueOf	Landroid/content/res/TypedArray;->peekValue
Ljava/lang/CharSequence;->toString	Landroid/os/Binder;->onTransact
Ljava/io/OutputStream;->write	Landroid/app/Notification;->setFullScreenIntent
Ljava/lang/String;->split	Ljava/util/HashSet;->remove
Landroid/os/SystemClock;->elapsedRealtime	Landroid/database/Cursor;->registerContentObserver
Ljava/lang/reflect/Method;->getParameterTypes	Ljava/lang/Thread;->interrupt
Landroid/content/Context;->startActivity	Landroid/content/IntentFilter;->addCategory
Ljava/lang/Class;->getConstructor	Landroid/graphics/drawable/AnimationDrawable;->isOneShot
Landroid/content/Intent;->getBooleanExtra	Landroid/os/Parcel;->createTypedArrayList
Ljava/io/File;->createNewFile	Landroid/content/res/Resources;->getText
Ljava/lang/Thread;->getName	Landroid/view/VelocityTracker;->computeCurrentVelocity

检测方法实例三：恶意APP



选择同样数目的特征，**粒子群算法**选择出的组合特征检测正确率要**高于正则化算法**。



粒子群算法准确率高，但是效率低，而正则化算法效率高，将两个算法结合，**首先利用正则化算法选择部分组合特征**，保证效率；**再利用粒子群算法进一步挖掘组合特征**，保证较高准确率。

检测方法实例三：恶意APP

不同机器学习分类器性能比较:

		NB	KNN	LR	SVM	DT	Adaboost
140 features by PSO	accuracy	0.9163	0.9130	0.9970	0.9988	0.9994	0.9994
	F1	0.8281	0.9156	0.9970	0.9988	0.9994	0.9994
280 features by PSO	accuracy	0.8200	0.9209	0.9994	1.000	1.000	1.000
	F1	0.8303	0.9239	0.9994	1.000	1.000	1.000
420 features by regularization	accuracy	0.8273	0.9355	0.9757	0.9769	0.9197	0.9642
	F1	0.8408	0.9376	0.9761	0.9773	0.9207	0.9650
All features	accuracy	0.7665	0.8765	0.9751	0.9811	1.000	1.000
	F1	0.7623	0.8742	0.9757	0.9819	1.000	1.000

试验结果：基于正则化和粒子群优化算法挖掘的组合特征能够提高Android恶意应用检测的性能；DT和Adaboost的检测正确率要优于其他算法。

问题和讨论

注：机器学习、群智能等方法的算法原理，后续课程会进行详解，本次课仅讨论检测原理。