
# 第二部分　访问控制

## 01.　口令破解与防护

厚 德 健 行

# 访问控制



- 访问控制有两个主要部分：认证和授权
- 认证:谁能去做?
  - 确定某个用户是否被允许访问
  - 机器对人进行认证
  - 机器对机器进行认证
- 授权: 是否被允许做某件事?
  - 一旦可以访问系统，那些行为是允许的?
  - 访问权限更细化的约束和限制
- 注意：“访问控制”常被用作“授权”的同义词

# 认证方法



- 基本问题：机器对用户怎样进行认证?
- 用户可以基于以下任何一点被机器认证：
  - 你所知的
    - 如：口令
  - 你所拥有的
    - 如：智能卡
  - 你本身的特征
    - 如：指纹

# 你所知的：口令



- 可以作为口令的：
  - PIN码
  - 身份证的后几位
  - 你童年的昵称
  - 你的生日
  - 宠物的名字
  - ……
- 为何口令如此流行？
- 成本: 口令是免费的
- 方便: 采用一个新口令比提供和配置一个新的智能卡要方便

# 口令的陷阱——搜狐全体员工遭“工资补助”诈骗



- 邮箱为何被盗？
- 为何可以群发邮件？

# 口令的陷阱——杭州某高校OA系统存在弱口令

浙江工业大学 ZHE JIANG UNIVERSITY OF TECHNOLOGY

编辑，否则保持在受保护视图中比较安全。 启用编辑(E)

已有账号单位用户名及初始密码

https://jd.citybrain.hangzhou.gov.cn/appointModule/home2

杭州城市大脑 HANGZHOU CITY BRAIN 首页 我要预约 预约管理 预约流程说明

本单位预约
函落款单位为本账号
直接预约

为其他单位预约
为友好单位代办
代办

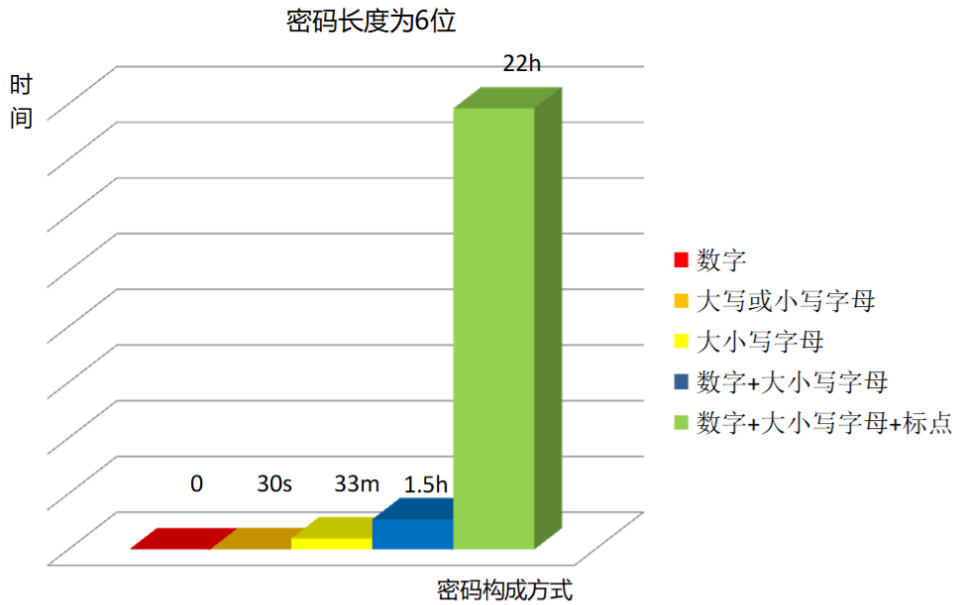
流程指引 温馨提醒 常见问

# 口令的陷阱——某校物联网管理平台存在弱口令

浙江工业大学 ZHE JIANG UNIVERSITY OF TECHNOLOGY

# 弱口令

- 弱口令的类型：
  - 简单的数字组合
  - 帐号相同的口令
  - 键盘上的临近键
  - 常见姓名
  - 终端设备出厂配置的通用口令
  - ……



# 通过口令发动网络攻击



- 攻击者的目标可能是...
  - 一个特定的账号
  - 某个系统上的任意账号
  - 任一系统上的任意账号
  - DoS攻击
- 通常的攻击路径
  - 外人 → 普通用户→ 管理员
  - 可能仅需要一个弱口令!

# 从口令开始防范网络攻击

浙江工业大学 ZHE JIANG UNIVERSITY OF TECHNOLOGY

- 良好的习惯
  - 在不同网站上尽量用不同的密码
  - 更换默认初始密码
  - 动态更换密码（rotation）
- 多因子认证
  - 所知道的、所拥有的、本身特征
  - 手持终端设备/生物特征
- 清理团队通用口令
  - 团队成员离职后，及时修改团队口令

# 多次重试口令



- 假设三次输入错误的口令，系统通常会被锁住，系统该锁多长时间?
  - 5秒钟
  - 5分钟
  - 或是直到管理员手动恢复服务
- 该锁多长时间?

# 口令的存储



- 将口令存放于文件中并不是个好办法
- 需要验证口令的有效性
- 解决方法:将hash过的口令存入文件
  - 存储 y = h(口令)
  - 将输入的口令经hash操作后与文件中的口令比较，以验证其有效性
  - 就算Trudy得到了口令文件，也得不到真正的口令
- 但是Trudy能寻找到一个快速的搜索方法
  - 他可以猜测可能的口令x，直到找到满足y = h(x)的x

# 字典攻击



- 攻击者对所有普通口令x做h(x)操作，并存入一个字典中。
- 假设攻击者可以访问包含hash后的口令的口令文件
  - 攻击者仅需要比较口令文件中的输入和她预计算的hash字典中的输入
  - 预计算的字典可以为每个口令文件重用

- 可以阻止这种攻击吗? 或者是让攻击者破解更难些?

# 口令的存储



- 将口令与salt值一起进行hash操作
- 给定口令为p，然后生成一个随机的salt值s，计算：

$$y = h(p, s)$$

  并将(s,y)存入口令文件中
- 注意: salt值s并不保密
- 易于验证口令
- Trudy必须重计算每个用户的hash值
  - 大大增加了Trudy的工作量!

# 口令破译中的数学问题



- 假设：
  - 所有的口令都是8个字符的长度，且每个字符有128种选择
    - 那就有$128^8 = 2^{56}$ 个可能的口令
  - 存在一个包含$2^{10}$个hash口令的口令文件
  - 攻击者有个包括$2^{20}$个普通口令的词典

从经验上讲，口令出现在攻击者字典中的可能性约为1/4

- 工作量由计算hash的次数来衡量

# 口令破译中的数学问题：情况一



- 在没有字典时寻找一个口令
  - 平均必须尝试$2^{56}/2 = 2^{55}$ 次
  - 类似于穷举密钥搜索方法
- 在这种情况下，salt值有帮助吗？
  - 没有任何帮助

# 口令破译中的数学问题：情况二



- 在有字典时寻找一个口令
- 使用salt值
  - 工作量约为：

$$1/4\ (2^{19}) + 3/4\ (2^{55}) = 2^{54.6}$$

  - 而实际上，要尝试字典中所有的单词，直到找不到口令为止
  - 工作量最多为$2^{20}$次，成功的可能性为1/4
- 如果没有使用salt值会怎么样呢？
  - 一次字典计算的次数： $2^{20}$
  - 预期工作量和上述的相同
  - 但是因为预计算字典哈希，“实际”攻击时是一帆风顺的

# 口令破译中的数学问题：情况三



- 寻找口令文件中1024个口令的任一个（没有字典的情况下）：
  - 假定文件中所有的$2^{10}$个口令都是不同的
  - 需要做$2^{55}$次不同的比较才能寻找到一个口令
- 若没有经过salt处理
  - 每个经过hash计算的值要比较$2^{10}$次
  - 以hash的次数来衡量，工作量为$2^{55}/2^{10}=2^{45}$
- 若经过salt处理
  - 预计工作量为$2^{55}$
  - 每次比较都需要一次hash计算

# 口令破译中的数学问题：情况四



- 寻找口令文件中1024个口令的任一个（存在字典时）：
  - 至少一个口令在字典中的概率是$1 - (3/4)^{1024} = 1$
  - 可以忽略掉字典中没包括口令的情况
- 若口令没经过salt处理
  - 有预计算所有的字典hash值，工作量为$2^{19}/2^{10} = 2^9$
- 若口令经过了salt处理，工作量小于$2^{22}$
  - 看本书或ppt中的备注（课后练习）
  - 近似的工作量：口令字典的大小/找出一个口令概率

# 口令的其他问题



- 要记下一大堆的口令
  - 导致口令的重复使用
  - 为何导致这样的问题?
- 谁是弱口令的受害者?
  - 登录口令与ATM的PIN码
- 没有修改默认口令
- 社会工程学
- 错误日志也许包含了“差不多”的口令
- 错误, 键登录, 间谍软件等.

# 小结



- 底线
- **口令易于被破译!**
  - 一个弱口令可能破坏系统的安全性
  - 用户可能选择不安全的口令
  - 社会工程学的攻击等.
- 坏人拥有口令后，无疑具有优势
- 数学工具对坏人起了帮助作用
- 口令是现实世界中最严重的安全问题之一
  - 并且将一直是一个大问题


# 关注我，下节内容更精彩：
# 02. 生物认证