【Web安全笔记】之【11.0 其他】

转载

AA8j 于 2020-12-23 18:33:51 发布 1587 收藏 2

分类专栏： Web安全笔记

原文链接：https://blog.csdn.net/zhj082/article/details/80531628


Web安全笔记 专栏收录该内容

9 篇文章 6 订阅

订阅专栏

文章目录

- 11.0 其他
  - 11.1 代码审计
    - 11.1.1 简介
    - 11.1.2 常用概念
      1. 输入
      2. 处理函数
      3. 危险函数
    - 11.1.3 自动化审计
      1. 危险函数匹配
      2. 控制流分析
      3. 基于图的分析
      4. 代码相似性比对
      5. 灰盒分析
    - 11.1.4 手工审计流程
      1. 获取代码，确定版本，尝试初步分析
      2. 基于审计工具进行初步分析
      3. 了解程序运行流程
         1) 文件加载方式
         2) 数据库连接方式
         3) 视图渲染
         4) SESSION处理机制
         5) Cache处理机制
      4. 账户体系
         1) Auth方式
         2) Pre-Auth的情况下可以访问的页面

3) 普通用户的帐号
4) 管理员账户默认密码
5) 账号体系
5. 根据漏洞类型查找Sink
1) SQLi
2) XSS
3) FILE
4) RCE
5) XXE
6) CSRF
7) SSRF
8) 反序列化
9) 变量覆盖
10) LDAP
11) XPath
12) Cookie伪造
6. 过滤
11.1.5 参考链接
11.2 WAF
11.2.1 简介
1. 概念
2. 常见功能
3. 布置位置
11.2.2 防护方式
11.2.3 扫描器防御
11.2.4 WAF指纹
11.2.5 绕过方式
1. 基于架构的绕过
2. 基于资源的绕过
3. 基于解析的绕过
4. 基于规则的绕过
11.2.6 参考链接
11.3 常见网络设备
11.3.1 防火墙
1. 简介
2. 主要功能
3. 下一代防火墙
11.3.2 IDS
1. 简介

2. 主要类型
11.3.3 安全隔离网闸
1. 简介
2. 主要功能
11.3.4 VPN设备
1. 简介
2. 常用技术
11.3.5 安全审计系统
1. 简介
11.3.6 参考链接
11.4 Unicode
11.4.1 基本概念
1. BMP
2. 码平面
3. Code Point
4. Code Unit
5. Surrogate Pair
6. Combining Character
7. BOM
11.4.2 编码方式
1. UCS-2
2. UTF-8
3. UTF-16
11.4.3 等价性问题
1. 简介
2. 标准等价
3. 兼容等价
4. 正规化
5. 正规形式
11.4.4 Tricks
11.4.5 安全问题
1. Visual Spoofing
2. Best Fit
3. Syntax Spoofing
4. Punycode Spoofs
5. Buffer Overflows
11.4.6 常见载荷
1. URL
2. SQL注入

3. XSS
4. 命令注入
5. 模板注入

11.4.7 参考链接

1. 官方文档
2. RFC
3. Tricks / Blogs

11.5 拒绝服务攻击

11.5.1 简介
11.5.2 UDP反射
11.5.3 TCP Flood
11.5.4 Shrew DDoS
11.5.5 Ping Of Death
11.5.6 Challenge Collapsar (CC)
11.5.7 慢速攻击
11.5.8 基于服务特性
11.5.9 常用的防护方式
11.5.10 参考链接

11.6 Docker

11.6.1 虚拟化技术与容器技术

1. 传统虚拟化技术
2. 容器技术

11.6.2 Docker

1. 基本概念
2. 组成
3. 数据
4. 网络

11.6.3 安全风险与安全机制

1. Docker安全基线
2. 内核命名空间/namespace
3. Control Group
4. 守护进程的攻击面
5. Capability
6. Seccomp

11.6.4 攻击面分析

1. 供应链安全
2. 虚拟化风险
3. 利用内核漏洞逃逸
4. 容器逃逸漏洞

5. 配置不当
6. 拒绝服务
7. 危险挂载
8. 攻击 Docker 守护进程
9. 其他CVE
11.6.5 安全加固
11.6.6 Docker 环境识别
1. Docker内
2. Docker外
11.6.7 参考链接
11.7 APT
11.7.1 简介
11.7.2 高级性（Advanced）
11.7.3 持续性（Persistent）
1. 侦查阶段
2. 初次入侵阶段
3. 权限提升阶段
4. 保持访问阶段
5. 横向扩展阶段
6. 攻击收益阶段
11.7.4 威胁性（Threat）
11.7.8 相关事件
11.7.9 IoC
11.7.10 参考链接
11.8 近源渗透
11.8.1 USB攻击
1. BadUSB
2. AutoRUN
3. USB Killer
4. 侧信道
5. HID攻击
11.8.2 Wi-Fi
1. 密码爆破
2. 信号压制
11.8.3 门禁
1. 电磁脉冲
2. IC卡
11.8.4 参考链接
11.9 常见术语

11.9.1 系统相关
11.9.2 网络相关
1. 网络协议
2. 路由系统
3. 网络应用
11.9.3 开发相关
11.9.4 安全相关
1. 安全开发
2. 安全策略
11.9.5 攻击相关
1. 漏洞类型
2. 攻击方式
11.9.6 防御相关
1. 防御技术
2. 防护设施
11.9.7 运维
11.9.8 认证

11.0 其他

11.1 代码审计

11.1.1 简介

代码审计是找到应用缺陷的过程。其通常有白盒审计、黑盒审计、灰盒审计等方式。白盒审计指通过对源代码的分析找到应用缺陷，黑盒审计通常不涉及到源代码，多使用模糊测试的方式，而灰盒审计则是黑白结合的方式。三种不同的测试方法有不同的优缺点。

11.1.2 常用概念

1. 输入

输入通常也称Source，Web应用的输入可以是请求的参数（GET、POST等）、上传的文件、Cookie、数据库数据等用户可控或者间接可控的地方。

例如PHP中的 `$_GET` / `$_POST` / `$_REQUEST` / `$_COOKIE` / `$_FILES` / `$_SERVER` 等，都可以作为应用的输入。

2. 处理函数

处理函数是对数据进行过滤或者编解码的函数，通常被称为Clean/Filter/Sanitizer。这些函数会对输入进行安全操作或过滤，为漏洞利用带来不确定性。

同样以PHP为例，这样的函数可能是 `mysqli_real_escape_string` / `htmlspecialchars` / `base64_encode` / `str_rot13` 等，也可能是应用自定义的过滤函数。

3. 危险函数

危险函数又常叫做Sink Call、漏洞点，是可能触发危险行为如文件操作、命令执行、数据库操作等行为的函数。

在PHP中，可能是 `include` / `system` / `echo` 等。

11.1.3 自动化审计

一般认为一个漏洞的触发过程是从输入经过过滤到危险函数的过程(Source To Sink)，而审计就是寻找这个链条的过程。常见的自动化审计方案有危险函数匹配、控制流分析等。

1. 危险函数匹配

白盒审计最常见的方式是通过搜寻危险函数与危险参数定位漏洞，比较有代表性的工具是Seay开发的审计工具。这种方法误报率相当高，这是因为这种方法没有对程序的流程进行深入分析，另一方面，这种方式通常是孤立地分析每一个文件，忽略了文件之间复杂的调用关系。

具体的说，这种方式在一些环境下能做到几乎无漏报，只要审计者有耐心，可以发现大部分的漏洞，但是在高度框架化的代码中，能找到的漏洞相对有限。

2. 控制流分析

在后来的系统中，考虑到一定程度引入AST作为分析的依据，在一定程度上减少了误报，但是仍存在很多缺陷。

而后，Dahse J等人设计了RIPS，该工具进行数据流与控制流分析，结合过程内与过程间的分析得到审计结果，相对危险函数匹配的方式来说误报率少了很多，但是同样的也增加了开销。

3. 基于图的分析

基于图的分析是对控制流分析的一个改进，其利用CFG的特性和图计算的算法，一定程度上简化了计算，比较有代表性的是微软的Semmle QL和NDSS 2017年发表的文章Efficient and Flexible Discovery of PHP Application Vulnerabilities。

4. 代码相似性比对

一些开发者会复制其他框架的代码，或者使用各种框架。如果事先有建立对应的漏洞图谱，则可使用相似性方法来找到漏洞。

5. 灰盒分析

基于控制流的分析开销较大，于是有人提出了基于运行时的分析方式，对代码进行Hook，当执行到危险函数时自动回溯输入，找到输入并判断是否可用。

这种方式解决了控制流分析实现复杂、计算路径开销大的问题，在判断过滤函数上也有一定的突破，但是灰盒的方式并不一定会触发所有的漏洞。fate0开发的prvd就是基于这种设计思路。

11.1.4 手工审计流程

1. 获取代码，确定版本，尝试初步分析

- 找历史漏洞信息
- 找应用该系统的实例
- 确定依赖库是否存在漏洞

2. 基于审计工具进行初步分析

3. 了解程序运行流程

1) 文件加载方式

- 类库依赖
- 是否加载waf

2) 数据库连接方式

- mysql/mysqli/pdo
- 是否开启预编译

3) 视图渲染

- XSS
- 模版注入

4) SESSION处理机制

- 文件
- 数据库
- 内存

5) Cache处理机制

- 文件cache可能写shell
- 数据库cache可能注入
- memcache

4. 账户体系

1) Auth方式

2) Pre-Auth的情况下可以访问的页面

3) 普通用户的帐号

- 能否可获取普通用户权限

4) 管理员账户默认密码

5) 账号体系

- 加密方式
- 爆破密码
- 重置漏洞
- 修改密码漏洞
    - 修改其他账号密码

5. 根据漏洞类型查找Sink

1) SQLi

- 全局过滤能否bypass
- 是否有直接执行SQL的地方
- SQL使用驱动，mysql/mysqli/pdo
    - 如果使用PDO，搜索是否存在直接执行的部分

2) XSS

- 全局bypass
- 视图渲染

3) FILE

- 查找上传功能点
- 上传下载覆盖删除
- 包含
  - LFI
  - RFI
  - 全局找include, require

4) RCE

5) XXE

6) CSRF

7) SSRF

8) 反序列化

9) 变量覆盖

10) LDAP

11) XPath

12) Cookie伪造

6. 过滤

找WAF过滤方式，判断是否可以绕过

11.1.5 参考链接

- rips
- prvd
- PHP运行时漏洞检测
- Backes M , Rieck K , Skoruppa M , et al. Efficient and Flexible Discovery of PHP Application Vulnerabilities[C]// IEEE European Symposium on Security & Privacy. IEEE, 2017.
- Dahse J. RIPS-A static source code analyser for vulnerabilities in PHP scripts[J]. Retrieved: February, 2010, 28: 2012.

11.2 WAF

11.2.1 简介

1. 概念

WAF（Web Application Firewall，Web应用防火墙）是通过执行一系列针对HTTP/HTTPS的安全策略来专门为Web应用提供加固的产品。

在市场上，有各种价格各种功能和选项的WAF。在一定程度上，WAF能为Web应用提供安全性，但是不能保证完全的安全。

2. 常见功能

- 检测异常协议，拒绝不符合HTTP标准的请求
- 对状态管理进行会话保护
- Cookies保护
- 信息泄露保护
- DDoS防护
- 禁止某些IP访问
- 可疑IP检查
- 安全HTTP头管理
  - X-XSS-Protection
  - X-Frame-Options
- 机制检测
  - CSRF token
  - HSTS

3. 布置位置

按布置位置，WAF可以分为云WAF、主机防护软件和硬件防护。云WAF布置在云上，请求先经过云服务器而后流向主机。主机防护软件需要主机预先安装对应软件，如mod_security、ngx-lua-waf等，对主机进行防护。硬件防护指流量流向主机时，先经过设备的清洗和拦截。

11.2.2 防护方式

WAF常用的方法有关键字检测、正则表达式检测、语法分析、行为分析、声誉分析、机器学习等。

基于正则的保护是最常见的保护方式。开发者用一些设定好的正则规则来检测载荷是否存在攻击性。基于正则的防护较为简单，因此存在一些缺点。例如只能应用于单次请求，而且正则很难应用到一些复杂的协议上。

基于语法的分析相对正则来说更快而且更准确，这种分析会把载荷按照语法解析成的符号组，然后在符号组中寻找危险的关键字。这种方式对一些载荷的变式有较好的效果，但是同样的，对解析器要求较高。

基于行为的分析着眼的范围更广一些，例如攻击者的端口扫描行为、目录爆破、参数测试或者一些其他自动化或者攻击的模式都会被纳入考虑之中。

基于声誉的分析可以比较好的过滤掉一些可疑的来源，例如常用的VPN、匿名代理、Tor节点、僵尸网络节点的IP等。

基于机器学习的WAF涉及到的范围非常广，效果也因具体实现和场景而较为多样化。

除了按具体的方法分，也可以根据白名单和黑名单的使用来分类。基于白名单的WAF适用于稳定的Web应用，而基于黑名单则适合处理已知问题。

11.2.3 扫描器防御

- 基于User-Agent识别
- 基于攻击载荷识别
- 验证码

11.2.4 WAF指纹

- 额外的Cookie
- 额外的Header
- 被拒绝请求时的返回内容
- 被拒绝请求时的返回响应码
- IP

11.2.5 绕过方式

1. 基于架构的绕过

- 站点在WAF后，但是站点可直连
- 站点在云服务器中，对同网段服务器无WAF

2. 基于资源的绕过

- 使用消耗大的载荷，耗尽WAF的计算资源
- 提供大量的无效参数

3. 基于解析的绕过

- 字符集解析不同
- 协议覆盖不全
  - POST的JSON传参 / `form-data` / `multipart/form-data`
- 协议解析不正确
- 站点和WAF对https有部分不一致
- WAF解析与Web服务解析不一致
  - 部分ASP+IIS会转换 `%u0065` 格式的字符
  - Apache会解析畸形Method
  - 同一个参数多次出现，取的位置不一样
  - HTTP Parameter Pollution (HPP)
  - HTTP Parameter Fragmentation (HPF)

4. 基于规则的绕过

等价替换

- 大小写变换 `select` => `sEleCt` `alert(1)`
- `<sCrIpt>alert(1)</script>`

字符编码

- URL编码
- 十六进制编码
- Unicode解析
- Base64
- HTML
- JSFuck
- 其他编码格式

等价函数

等价变量

关键字拆分

字符串操作

字符干扰

空字符

- NULL (x00)
- 空格
- 回车 (x0d)
- 换行 (x0a)
- 垂直制表 (x0b)
- 水平制表 (x09)
- 换页 (x0c)

注释

特殊符号

- 注释符
- 引号（反引号、单引号、双引号）

利用服务本身特点

- 替换可疑关键字为空 `selselectect` => `select`

少见特性未在规则列表中

11.2.6 参考链接

- WAF攻防研究之四个层次Bypass WAF
- 我的WafBypass之道 SQL注入篇
- WAF through the eyes of hackers

11.3 常见网络设备

11.3.1 防火墙

1. 简介

防火墙指的是一个有软件和硬件设备组合而成、在内部网和外部网之间、专用网与公共网之间的界面上构造的保护屏障。它可通过监测、限制、更改跨越防火墙的数据流，尽可能地对外部屏蔽网络内部的信息、结构和运行状况，以此来实现网络的安全保护。

防火墙可以分为网络层防火墙和应用层防火墙。网络层防火墙基于源地址和目的地址、应用、协议以及每个IP包的端口来作出通过与否的判断。应用层防火墙针对特别的网络应用服务协议即数据过滤协议，并且能够对数据包分析并形成相关的报告。

2. 主要功能

- 过滤进、出网络的数据
- 防止不安全的协议和服务
- 管理进、出网络的访问行为
- 记录通过防火墙的信息内容
- 对网络攻击进行检测与警告
- 防止外部对内部网络信息的获取
- 提供与外部连接的集中管理

3. 下一代防火墙

主要是一款全面应对应用层威胁的高性能防火墙。可以做到智能化主动防御、应用层数据防泄漏、应用层洞察与控制、威胁防护等特性。 下一代防火墙在一台设备里面集成了传统防火墙、IPS、应用识别、内容过滤等功能既降低了整体网络安全系统的采购投入，又减去了多台设备接入网络带来的部署成本，还通过应用识别和用户管理等技术降低了管理人员的维护和管理成本。

11.3.2 IDS

1. 简介

入侵检测即通过从网络系统中的若干关键节点收集并分析信息，监控网络中是否有违反安全策略的行为或者是否存在入侵行为。入侵检测系统通常包含3个必要的功能组件：信息来源、分析引擎和响应组件。

信息收集包括收集系统、网络、数据及用户活动的状态和行为。入侵检测利用的信息一般来自：系统和网络日志文件、非正常的目录和文件改变、非正常的程序执行这三个方面。

分析引擎对收集到的有关系统、网络、数据及用户活动的状态和行为等信息，是通过模式匹配、统计分析和完整性分析这三种手段进行分析的。前两种用于实时入侵检测，完整性分析用于事后分析。

告警与响应根据入侵性质和类型，做出相应的告警与响应。

2. 主要类型

IDS可以分为基于主机的入侵检测系统(HIDS)和基于网络的入侵检测系统(NIDS)。

基于主机的入侵检测系统是早期的入侵检测系统结构，通常是软件型的，直接安装在需要保护的主机上。其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日志发现可疑事件。

这种检测方式的优点主要有：信息更详细、误报率要低、部署灵活。这种方式的缺点主要有：会降低应用系统的性能；依赖于服务器原有的日志与监视能力；代价较大；不能对网络进行监测；需安装多个针对不同系统的检测系统。

基于网络的入侵检测方式是目前一种比较主流的监测方式，这类检测系统需要有一台专门的检测设备。检测设备放置在比较重要的网段内，不停地监视网段中的各种数据包，而不再是只监测单一主机。它对所监测的网络上每一个数据包或可疑的数据包进行特征分析，如果数据包与产品内置的某些规则吻合，入侵检测系统就会发出警报，甚至直接切断网络连接。目前，大部分入侵检测产品是基于网络的。

这种检测技术的优点主要有：能够检测那些来自网络的攻击和超过授权的非法访问；不需要改变服务器等主机的配置，也不会影响主机性能；风险低；配置简单。其缺点主要是：成本高、检测范围受局限；大量计算，影响系统性能；大量分析数据流，影响系统性能；对加密的会话过程处理较难；网络流速高时可能会丢失许多封包，容易让入侵者有机可乘；无法检测加密的封包；对于直接对主机的入侵无法检测出。

11.3.3 安全隔离网闸

1. 简介

安全隔离网闸是使用带有多种控制功能的固态开关读写介质连接两个独立网络系统的信息安全设备。由于物理隔离网闸所连接的两个独立网络系统之间，不存在通信的物理连接、逻辑连接、信息传输命令、信息传输协议，不存在依据协议的信息包转发，只有数据文件的无协议“摆渡”，且对固态存储介质只有“读”和“写”两个命令。所以，物理隔离网闸从物理上隔离、阻断了具有潜在攻击可能的一切连接，使攻击者无法入侵、无法攻击、无法破坏，实现了真正的安全。

2. 主要功能

阻断网络的直接物理连接：物理隔离网闸在任何时刻都只能与非可信网络和可信网络上之一相连接，而不能同时与两个网络连接。

阻断网络的逻辑连接：物理隔离网闸不依赖操作系统、不支持TCP/IP协议。两个网络之间的信息交换必须将TCP/IP协议剥离，将原始数据通过P2P的非TCP/IP连接方式，通过存储介质的“写入”与“读出”完成数据转发。

安全审查：物理隔离网闸具有安全审查功能，即网络在将原始数据“写入”物理隔离网闸前，根据需要对原始数据的安全性进行检查，把可能的病毒代码、恶意攻击代码过滤掉。

原始数据无危害性：物理隔离网闸转发的原始数据，不具有攻击或对网络安全有害的特性。

管理和控制功能：建立完善的日志系统。

根据需要建立数据特征库：在应用初始化阶段，结合应用要求，提取应用数据的特征，形成用户特有的数据特征库，作为运行过程中数据校验的基础。当用户请求时，提取用户的应用数据，抽取数据特征和原始数据特征库比较，符合原始特征库的数据请求进入请求队列，不符合的返回用户，实现对数据的过滤。

根据需要提供定制安全策略和传输策略的功能：用户可以自行设定数据的传输策略，如：传输单位（基于数据还是基于任务）、传输间隔、传输方向、传输时间、启动时间等。

支持定时/实时文件交换；支持支持单向/双向文件交换；支持数字签名、内容过滤、病毒检查等功能。

11.3.4 VPN设备

1. 简介

虚拟专用网络指的是在公用网络上建立专用网络的技术。之所以称为虚拟网主要是因为整个VPN网络的任意两个节点之间的连接并没有传统专网所需的端到端的物理链路，而是架构在公用网络服务商所提供的网络平台之上的逻辑网络，用户数据在逻辑链路中传输。

2. 常用技术

MPLS VPN：是一种基于MPLS技术的IP VPN，是在网络路由和交换设备上应用MPLS（多协议标记交换）技术，简化核心路由器的路由选择方式，利用结合传统路由技术的标记交换实现的IP虚拟专用网络（IP VPN）。MPLS优势在于将二层交换和三层路由技术结合起来，在解决VPN、服务分类和流量工程这些IP网络的重大问题时具有很优异的表现。因此，MPLS VPN在解决企业互连、提供各种新业务方面也越来越被运营商看好，成为在IP网络运营商提供增值业务的重要手段。MPLS VPN又可分为二层MPLS VPN（即MPLS L2 VPN）和三层MPLS VPN（即MPLS L3 VPN）。

SSL VPN：是以HTTPS（SecureHTTP，安全的HTTP，即支持SSL的HTTP协议）为基础的VPN技术，工作在传输层和应用层之间。SSL VPN充分利用了SSL协议提供的基于证书的身份认证、数据加密和消息完整性验证机制，可以为应用层之间的通信建立安全连接。SSL VPN广泛应用于基于Web的远程安全接入，为用户远程访问公司内部网络提供了安全保证。

IPSecVPN是基于IPSec协议的VPN技术，由IPSec协议提供隧道安全保障。IPSec是一种由IETF设计的端到端的确保基于IP通讯的数据安全性的机制。它为Internet上传输的数据提供了高质量的、可互操作的、基于密码学的安全保证。

11.3.5 安全审计系统

1. 简介

网络安全审计系统针对互联网行为提供有效的行为审计、内容审计、行为报警、行为控制及相关审计功能。从管理层面提供互联网的有效监督，预防、制止数据泄密。满足用户对互联网行为审计备案及安全保护措施的要求，提供完整的上网记录，便于信息追踪、系统安全管理和风险防范。

11.3.6 参考链接

- 网络安全设备

11.4 Unicode

11.4.1 基本概念

1. BMP

BMP (Basic Multilingual Plane)，译作基本多文种平面，是Unicode中的一个编码区块。

2. 码平面

Unicode编码点分为17个平面（plane），每个平面包含2^16（即65536）个码位。17个平面的码位可表示为从U+xx0000到U+xxFFFF，其中xx表示十六进制值从0016到1016，共计17个平面。

3. Code Point

Code Point也被称作Code Position，译作码位或编码位置，是指组成代码空间的数值。

4. Code Unit

指某种 Unicode 编码方式里编码一个 Code Point 需要的最少字节数，比如 UTF-8 需要最少一个字节，UTF-16 最少两个字节，UCS-2 两个字节，UCS-4 和 UTF-32 四个字节。

5. Surrogate Pair

Surrogate Pair 是用于 UTF-16 的以向后兼容 UCS-2 的，做法是取 UCS-2 范围里的 0xD800~0xDBFF (称为 high surrogates) 和 0xDC00~0xDFFF (称为 low surrogates) 的码位，一个 high surrogate 接一个 low surrogate 拼成四个字节表示超出 BMP 的字符，两个 surrogate range 都是 1024 个码位，所以 surrogate pair 可以表达 1024 x 1024 = 1048576 = 0x100000 个字符。

6. Combining Character

例如 `Hål̈ö` 含有重音符号之类的字符，进行组合会使用大量的码位。所以这种字符多用组合的方式来实现。

7. BOM

字节顺序标记（byte-order mark，BOM）是一个有特殊含义的统一码字符，码点为 `U+FEFF` 。当以UTF-16或UTF-32来将UCS字符所组成的字符串编码时，这个字符被用来标示其字节序。常被用于区分是否为UTF编码。

11.4.2 编码方式

1. UCS-2

UCS-2 (2-byte Universal Character Set)是一种定长的编码方式，UCS-2仅仅简单的使用一个16位码元来表示码位，也就是说编码范围在0到0xFFFF的码位范围内。

2. UTF-8

UTF-8（8-bit Unicode Transformation Format）是一种针对Unicode的可变长度字符编码，也是一种前缀码。它可以用一至四个字节对Unicode字符集中的所有有效编码点进行编码，属于Unicode标准的一部分。编码方式如下

码点的位数	码点起值	码点终值	字节序列	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
7	U+0000	U+007F	1	0xxxxxxx					
11	U+0080	U+07FF	2	110xxxxx	10xxxxxx				
16	U+0800	U+FFFF	3	1110xxxx	10xxxxxx	10xxxxxx			
21	U+10000	U+1FFFFF	4	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx		
26	U+200000	U+3FFFFFF	5	111110xx	10xxxxxx	10xxxxxx	10xxxxxx	10xxxxxx	
31	U+4000000	U+7FFFFFFF	6	1111110x	10xxxxxx	10xxxxxx	10xxxxxx	10xxxxxx	10xxxxxx

3. UTF-16

UTF-16 (16-bit Unicode Transformation Format)是UCS-2的拓展，用一个或者两个16位的码元来表示码位，可以对0到0x10FFFF的码位进行编码。

11.4.3 等价性问题

1. 简介

Unicode（统一码）包含了许多特殊字符，为了使得许多现存的标准能够兼容，提出了Unicode等价性（Unicode equivalence）。在字符中，有些在功能上会和其它字符或字符序列等价。因此，Unicode将一些码位序列定义成相等的。

Unicode提供了两种等价概念：标准等价和兼容等价。前者是后者的一个子集。例如，字符n后接着组合字符 ~ 会（标准和兼容）等价于Unicode字符ñ。而合字ff则只有兼容等价于两个f字符。

Unicode正规化是文字正规化的一种形式，是指将彼此等价的序列转成同一列序。此序列在Unicode标准中称作正规形式。

对于每种等价概念，Unicode又定义两种形式，一种是完全合成的，一种是完全分解的。因此，最后会有四种形式，其缩写分别为：NFC、NFD、NFKC、NFKD。对于Unicode的文字处理程序而言，正规化是很重要的。因为它影响了比较、搜索和排序的意义。

2. 标准等价

统一码中标准等价的基础概念为字符的组成和分解的交互使用。合成指的是将简单的字符合并成较少的预组字符的过程，如字符n和组合字符 ~ 可以组成统一码ñ。分解则是反向过程，即将预组字符变回部件。

标准等价是指保持视觉上和功能上的等价。例如，含附加符号字母被视为和分解后的字母及其附加符号是标准等价。换句话说，预组字符‘ü’和由‘u’及 ‘¨’所组成的序列是标准等价。相似地，统一码统合了一些希腊附加符号和外观与附加符号类似的标点符号。

3. 兼容等价

兼容等价的范围较标准等价来得广。如果序列是标准等价的话就会是兼容等价，反之则未必对。兼容等价更关注在于纯文字的等价，并把一些语义上的不同形式归结在一起。

例如，上标数字和其所使用的数字是兼容等价，但非标准等价。其理由为下标和上标形式虽然在某些时侯属于不同意义，但若应用程序将他们视为一样也是合理的（虽然视觉上可区分）。如此，在统一码富文件中，上标和下标就可以以比较不累赘地方式出现（见下一节）。

全角和半角的片假名也是一种兼容等价但不是标准等价。如同合字和其部件序列。其只有视觉上但没有语义上的区别。换句话说，作者通常没有特别宣称使用合字是一种意思，而不使用是另一种意思。相对地，这尽限于印刷上的选择。

文字处理软件在实现统一码字符串的搜索和排序时，须考虑到等价性的存在。如果没有此特性的话，用户在搜索时将无法找到在视觉上无法区分的字形。

统一码提供了一个标准的正规化算法，可将所有相同的序列产生一个唯一的序列。 其等价准绳可以为标准的（NF）或兼容的（NFK）。 既然可以任意选择等价类中的元素，对每一个等价标准有多个标准形式也是有可能的。 统一码为每一种等价准绳分别提供两种正规形式：合成用的NFC和NFKC以及分解用的NFD和NFKD。 而不论是组合的或分解的形式，都会使用标准顺序，以此限制正规形式只有唯一形式。

4. 正规化

为了比较或搜索统一码字符串，软件可以使用合成或分解形式其中之一。 只要被比较或搜索的字符串使用的形式是相同的，哪种选择都没关系。 另一方面，等价概念的选择则会影响到搜索结果。 譬如，有些合字如ffi（U+FB03）、罗马数字如Ⅸ（U+2168），甚至是上标数字如 （U+2075）有其个别统一码码位。 标准正规形式并不会影响这些结果。 但兼容正规形式会分解ffi成f、f、i。所以搜索U+0066（f）时，在NFKC中会成功，但在NFC则会失败。 同样地有在预组罗马数字Ⅸ 中搜索拉丁字母I（U+0049）。类似地，“ ”会转成“5”。

对于浏览器，将上标转换成到基下划线未必是好的，因为上标的信息会因而消失。为了允许这种不同，统一码字符数据库句含了兼容格式标签，其提供了兼容转换的细节。在合字的情况下，这个标签只是 `，而在上标的情况下则为` 。丰富文件格式如超文本置标语言则会使用兼容标签。例如，HTML使用自定义标签来将“5”放到上标位置。

5. 正规形式

- NFD Normalization Form Canonical Decomposition 以标准等价方式来分解
- NFC Normalization Form Canonical Composition 以标准等价方式来分解，然后以标准等价重组之。若是singleton的话，重组结果有可能和分解前不同。
- NFKD Normalization Form Compatibility Decomposition 以兼容等价方式来分解NFKC
- Normalization Form Compatibility Composition 以兼容等价方式来分解，然后以标准等价重组之

11.4.4 Tricks

- 部分语言的长度并不是字符的长度，一个UTF-16可能是两位。
- 部分语言在翻转UTF-16等多字节编码时，会处理错误。

11.4.5 安全问题

1. Visual Spoofing

例如baidu.com(此处的a为u0430)和baidu.com(此处的a为x61)视觉上相同，但是实际上指向两个不同的域名。

b ａ idu.com(此处的a为uff41)和baidu.com(此处的a为x61)有一定的不同，但是指向两个相同的域名。

这种现象可以引起一些Spoofing或者WAF Bypass的问题。

2. Best Fit

如果两个字符前后编码不同，之前的编码在之后的编码没有对应，程序会尝试找最佳字符进行自动转换。

当宽字符变成了单字节字符，字符编码会有一定的变化。

这种现象可能引起一些WAF Bypass。

3. Syntax Spoofing

以下四个Url在语法上看来是没问题的域名，但是用来做分割的字符并不是真正的分割字符，而是U+2044(／)，可以导致一些UI欺骗的问题。

- http://macchiato.com/x.bad.com macchiato.com/x bad.com
- http://macchiato.com?x.bad.com macchiato.com?x bad.com
- http://macchiato.com.x.bad.com macchiato.com.x bad.com
- http://macchiato.com#x.bad.com macchiato.com#x bad.com

4. Punycode Spoofs

- http://藕蔼護蕸.com http://xn--google.com
- http://曦.com http://xn--cnn.com
- http://岍岜岜岅吻峉.com http://xn--citibank.com

有些浏览器会直接显示puncode，但是也可以借助这种机制实现UI Spooof。

5. Buffer Overflows

在编码转换的时候，有的字符会变成多个字符，如Fluß → FLUSS → fluss这样可能导致BOF。

11.4.6 常见载荷

1. URL

- ‥ (U+2025)
- ： (U+FE30)
- 。 (U+3002)
- ⓐ (U+24EA)
- ／ (U+FF0F)
- ｐ (U+FF50)
- ʰ (U+02B0)
- ª (U+00AA)

2. SQL注入

- `'` (U+FF07)
- `"` (U+FF02)
- `-` (U+FE63)

3. XSS

- `＜` (U+FF1C)
- `"` (U+FF02)

4. 命令注入

- `&` (U+FF06)
- `|` (U+FF5C)

5. 模板注入

- `｛` (U+FE5B)
- `［` (U+FF3B)

11.4.7 参考链接

1. 官方文档

- Unicode equivalence
- Unicode Normalization Forms
- Unicode Security Considerations

2. RFC

- RFC 3629 UTF-8, a transformation format of ISO 10646
- RFC 2044 UTF-8, a transformation format of ISO 10646
- RFC 2279 UTF-8, a transformation format of ISO 10646

3. Tricks / Blogs

- IDN homograph attack
- Black Hat Unicode Security
- Request encoding to bypass web application firewalls
- domain hacks with unusual unicode characters
- 其实你并不懂 Unicode

11.5 拒绝服务攻击

11.5.1 简介

DoS（Denial of Service）指拒绝服务，是一种常用来使服务器或网络瘫痪的网络攻击手段。

在平时更多提到的是分布式拒绝服务（DDoS，Distributed Denial of Service） 攻击，该攻击是指利用足够数量的傀儡计算机产生数量巨大的攻击数据包，对网络上的一台或多台目标实施DDoS攻击，成倍地提高威力，从而耗尽受害目标的资源，迫使目标失去提供正常服务的能力。

11.5.2 UDP反射

基于UDP文的反射DDoS攻击是拒绝服务攻击的一种形式。攻击者不直接攻击目标，而是利用互联网中某些开放的服务器，伪造被攻击者的地址并向该服务器发送基于UDP服务的特殊请求报文，使得数倍于请求报文的数据被发送到被攻击IP，从而对后者间接形成DDoS攻击。

常用于DoS攻击的服务有:

- NTP
- DNS
- SSDP
- Memcached

其中DNS攻击主要是指DNS Request Flood、DNS Response Flood、虚假源+真实源DNS Query Flood、权威服务器攻击和Local服务器攻击。

11.5.3 TCP Flood

TCP Flood是一种利用TCP协议缺陷的攻击，这种方式通过伪造IP向攻击服务器发送大量伪造的TCP SYN请求，被攻击服务器回应握手包后（SYN+ACK），因为伪造的IP不会回应之后的握手包，服务器会保持在SYN_RECV状态，并尝试重试。这会使得TCP等待连接队列资源耗尽，正常业务无法进行。

11.5.4 Shrew DDoS

Shrew DDoS利用了TCP的重传机制，调整攻击周期来反复触发TCP协议的RTO，达到攻击的效果。其数据包以固定的、恶意选择的慢速时间发送，这种模式能够将TCP流量限制为其理想速率的一小部分，同时以足够低的平均速率进行传输以避免检测。

现代操作系统已经对TCP协议进行了相应的修改，使得其不受影响。

11.5.5 Ping Of Death

在正常情况下不会存在大于65536个字节的ICMP包，但是报文支持分片重组机制。通过这种方式可以发送大于65536字节的ICMP包并在目标主机上重组，最终会导致被攻击目标缓冲区溢出，引起拒绝服务攻击。

现代操作系统已经对这种攻击方式进行检查，使得其不受影响。

11.5.6 Challenge Collapsar (CC)

CC攻击是一种针对资源的DoS攻击，攻击者通常会常用请求较为消耗服务器资源的方式来达到目的。

CC攻击的方式有很多种，常见的攻击可以通过大量访问搜索页、物品展示页等消耗大的功能来实现。部分HTTP服务器也可通过上传超大文件、发送大量且复杂的参数的请求来实现攻击。

11.5.7 慢速攻击

HTTP慢速攻击是由Wong Onn Chee 和 Tom Brennan在2012年的OWASP大会上正式披露，用低速发包来消耗服务器资源以达到拒绝服务的目的。

慢速攻击分为 Slow headers / Slow body / Slow read 三种攻击方式。Slow headers 一直不停的慢速发送HTTP头部，消耗服务器的连接和内存资源。Slow body 发送一个 Content-Length 很大的 HTTP POST请求，每次只发送很少量的数据，使该连接一直保持存活。Slow read以很低的速度读取Response。

11.5.8 基于服务特性

- 压缩包解压
  - 巨大的0字节的压缩包
- 读文件
  - 读 `/dev/urandom` 等无限制的文件
- 受限制的反序列化
  - 反序列化巨大的数组
- 正则解析
  - 消耗巨大的回溯表达式

11.5.9 常用的防护方式

- 基于特定攻击的指纹检测攻击，对相应流量进行封禁/限速操作
- 对正常流量进行建模，对识别出的异常流量进行封禁/限速操作
- 基于IP/端口进行综合限速策略
- 基于地理位置进行封禁/限速操作

11.5.10 参考链接

- linux academy dos
- slowhttptest Application Layer DoS attack simulator
- Slowloris wiki

11.6 Docker

11.6.1 虚拟化技术与容器技术

1. 传统虚拟化技术

传统虚拟化技术通过添加hypervisor层，虚拟出网卡，内存，CPU等虚拟硬件，再在其上建立客户机，每个客户机都有自己的系统内核。传统虚拟化技术以虚拟机为管理单元，各虚拟机拥有独立的操作系统内核，不共用宿主机的软件系统资源，因此具有良好的隔离性，适用于云计算环境中的多租户场景。

2. 容器技术

容器技术可以看作一种轻量级的虚拟化方式，容器技术在操作系统层进行虚拟化，可在宿主机内核上运行多个虚拟化环境。相比于传统的应用测试与部署，容器的部署无需预先考虑应用的运行环境兼容性问题；相比于传统虚拟机，容器无需独立的操作系统内核就可在宿主机中运行，实现了更高的运行效率与资源利用率。

11.6.2 Docker

Docker是目前最具代表性的容器平台之一，具有持续部署与测试、跨云平台支持等优点。在基于Kubernetes等容器编排工具实现的容器云环境中，通过对跨主机集群资源的调度，容器云可提供资源共享与隔离、容器编排与部署、应用支撑等功能。

1. 基本概念

Docker有三个基本概念，镜像（Image）、容器（Container）、仓库（Repository）。镜像是一个只读的模版，由一组文件系统通过Union FS技术组成。

镜像是静态的定义，容器是从镜像创建的运行实例。容器的本质是进程，拥有自己独立的命名空间。

仓库（Repository） 是集中存放镜像文件的场所，用于存储、分发镜像。

容器可以被启动、开始、停止、删除，每个容器都是相互隔离的，可以把容器看做是一个简易版的 Linux 环境（包括root用户权限、进程空间、用户空间和网络空间等）和运行在其中的应用程序。

2. 组成

Docker引擎由如下主要组件构成：Docker客户端（Docker Client）、Docker守护进程（Docker daemon）、containerd以及runc，它们共同负责容器的创建和运行。

Docker Client是和Docker Daemon建立通信客户端，Docker Client可以通过http/unix socket等方式Daemon建立通信。

Docker Daemon在宿主机运行，作为服务端接受来自客户端的请求，主要功能包括镜像管理、镜像构建、REST API、身份验证、安全、核心网络以及编排。Docker daemon通过位于 `/var/run/docker.sock` 的本地 IPC/Unix socket 来实现Docker远程API，默认非TLS网络端口为2375，TLS默认端口为2376。

3. 数据

Docker的数据主要分为持久化和非持久化数据，默认情况下非持久化存储是自动创建生命周期与容器相同，删除容器也会删除非持久化数据，在Linux环境下，非持久化数据默认存储于 `/var/lib/docker/` 下。

4. 网络

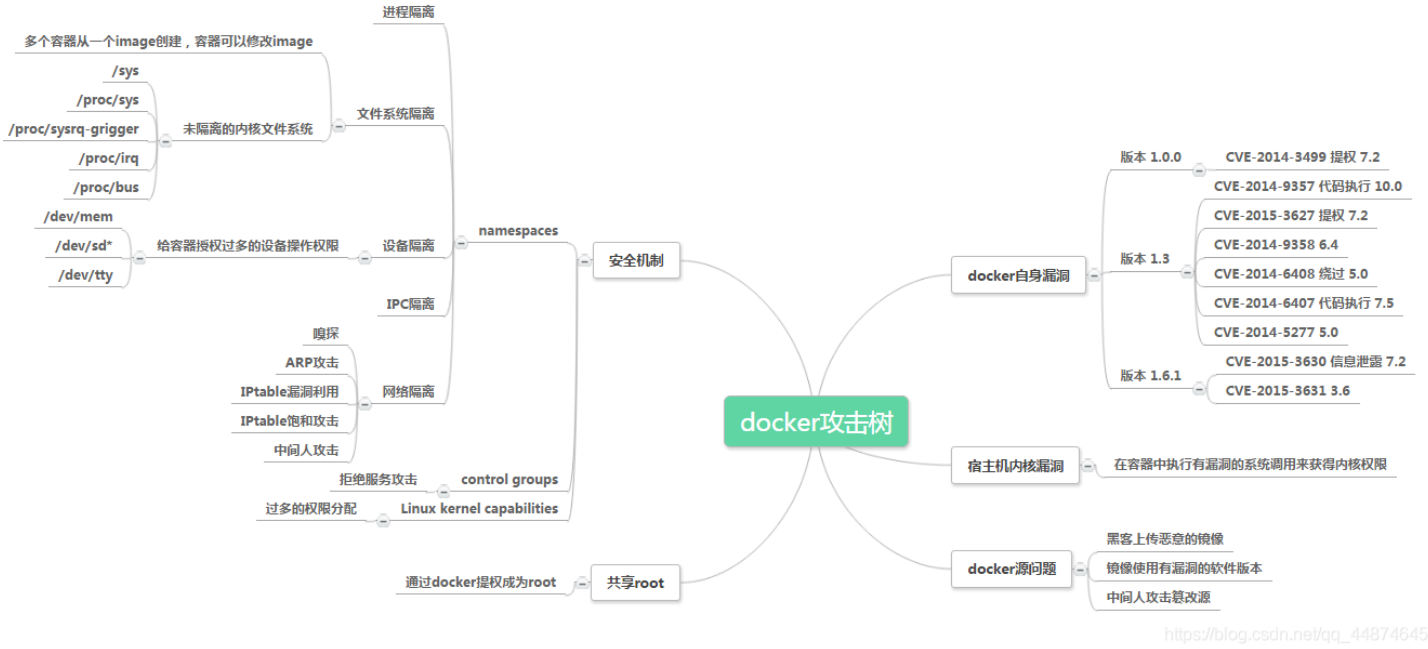
Docker网络架构源自一种叫作容器网络模型的方案，主要由CNM、Libnetwork、网络驱动构程。

11.6.3 安全风险与安全机制

在考虑Docker安全性的时候主要考虑以下几点

- 内核本身的安全性及其对命名空间和cgroups的支持
- Docker守护进程本身的攻击面
- 内核的“强化”安全功能以及它们如何与容器进行交互

1. Docker安全基线



2. 内核命名空间/namespace

Docker容器与LXC容器非常相似，并且具有相似的安全特性。当使用docker运行启动容器时，Docker会在后台为容器创建一组命名空间和控制组。

命名空间提供了一个最直接的隔离形式：在容器中运行的进程看不到或者无法影响在另一个容器或主机系统中运行的进程。

每个容器也有自己的网络堆栈，这意味着一个容器不能获得对另一个容器的套接字或接口的特权访问。当然，如果主机系统相应设置，容器可以通过各自的网络接口交互。如果为容器指定公共端口或使用链接时，容器之间允许IP通信。

它们可以相互ping通，发送/接收UDP数据包，并建立TCP连接，但是如果需要可以限制它们。从网络体系结构的角度来看，给定Docker主机上的所有容器都位于网桥接口上。这意味着它们就像通过普通的以太网交换机连接的物理机器一样。

3. Control Group

控制组是Linux容器的另一个关键组件，主要作用是实施资源核算和限制。

Cgroup 提供了许多有用的度量标准，但也有助于确保每个容器都能获得公平的内存，CPU和磁盘I/O; 更重要的是单个容器不能通过耗尽资源的方式来降低系统的性能。

因此，尽管 Cgroup 不能阻止一个容器访问或影响另一个容器的数据和进程，但它们对于抵御一些拒绝服务攻击是至关重要的。它们对于多租户平台尤其重要，例如公共和私人PaaS，即使在某些应用程序开始行为不当时也能保证一致的正常运行时间（和性能）。

4. 守护进程的攻击面

使用Docker运行容器意味着运行Docker守护进程，而这个守护进程当前需要root权限，因此，守护进程是需要考虑的一个地方。

首先，只有受信任的用户才能被允许控制Docker守护进程。具体来说，Docker允许您在Docker主机和访客容器之间共享一个目录;它允许你这样做而不限制容器的访问权限。这意味着可以启动一个容器，其中/host目录将成为主机上的/目录，容器将能够不受任何限制地改变主机文件系统。

这具有很强的安全意义：例如，如果通过Web服务器测试Docker以通过API配置容器，则应该更加仔细地进行参数检查，以确保恶意用户无法传递制作的参数，从而导致Docker创建任意容器。

守护进程也可能容易受到其他输入的影响，例如从具有docker负载的磁盘或从具有docker pull的网络加载映像。

最终，预计Docker守护进程将运行受限特权，将操作委托给审核良好的子进程，每个子进程都有自己的（非常有限的）Linux功能范围，虚拟网络设置，文件系统管理等。也就是说，很可能，Docker引擎本身的部分将在容器中运行。

5. Capability

默认情况下，Docker采用Capability机制来实现用户在以root身份运行容器的同时，限制部分root的操作。

在大多数情况下，容器不需要真正的root权限。因此，Docker可以运行一个Capability较低的集合，这意味着容器中的root比真正的root要少得多。例如：

- 否认所有挂载操作
- 拒绝访问原始套接字（防止数据包欺骗）
- 拒绝访问某些文件系统操作，如创建新的设备节点，更改文件的所有者或修改属性（包括不可变标志）
- 拒绝模块加载
- 其他

这意味着，即使入侵者在容器内获取root权限，进一步攻击也会困难很多。默认情况下，Docker使用白名单而不是黑名单，去除了所有非必要的功能。

6. Seccomp

Docker使用Seccomp来限制容器对宿主机内核发起的系统调用。

11.6.4 攻击面分析

1. 供应链安全

在构建Dockerfile的过程中，即使是使用排名靠前的来源，也可能存在CVE漏洞、后门、镜像污染等问题。

2. 虚拟化风险

虽然Docker通过命名空间进行了文件系统资源的基本隔离，但仍有 `/sys` 、 `/proc/sys` 、 `/proc/bus` 、 `/dev` 、 `time` 、 `syslog` 等重要系统文件目录和命名空间信息未实现隔离，而是与宿主机共享相关资源。

3. 利用内核漏洞逃逸

- CVE-2016-5195

4. 容器逃逸漏洞

- CVE-2019-14271 Docker cp
- CVE-2019-13139 Docker build code execution
- CVE-2019-5736 runC
  - Docker Version < 18.09.2
  - Version <= 1.0-rc6
- CVE-2018-18955

5. 配置不当

- 开启privileged
- 挂载宿主机敏感目录
- 配置cap不当
  - `--cap-add=SYS_ADMIN`
- 绕过namespace
  - `--net=host`
  - `--pid=host`
  - `--ipc=host`

6. 拒绝服务

- CPU耗尽
- 内存耗尽
- 存储耗尽
- 网络资源耗尽

7. 危险挂载

- 挂载 `/var/run/docker.sock`
- 挂载宿主机 `/dev` `/proc` 等危险目录

8. 攻击 Docker 守护进程

虽然 Docker 容器具有很强的安全保护措施，但是 Docker 守护进程本身并没有被完善的保护。Docker 守护进程本身默认由 root 用户运行，并且该进程本身并没有使用 Seccomp 或者 AppArmor 等安全模块进行保护。这使得一旦攻击者成功找到漏洞控制 Docker 守护进程进行任意文件写或者代码执行，就可以顺利获得宿主机的 root 权限而不会受到各种安全机制的阻碍。值得一提的是，默认情况下 Docker 不会开启 User Namespace 隔离，这也意味着 Docker 内部的 root 与宿主机 root 对文件的读写权限相同。这导致一旦容器内部 root 进程获取读写宿主机文件的机会，文件权限将不会成为另一个问题。这一点在 CVE-2019-5636 利用中有所体现。

9. 其他CVE

- CVE-2014-5277
- CVE-2014-6408
- CVE-2014-9357
- CVE-2014-9358
- CVE-2015-3627
- CVE-2015-3630

11.6.5 安全加固

- 最小安装
  - 删除所有开发工具（编译器等）
- 更新系统源
- 启用 AppArmor
- 启用 SELinux
- 限制运行容器的内核功能
- 移除依赖构建
- 配置严格的网络访问控制策略
- 不使用root用户启动docker
- 不以privileged特权模式运行容器
- 控制资源
  - CPU Share
  - CPU 核数
  - 内存资源
  - IO 资源
  - 磁盘资源
  - 硬件资源
- 使用安全的基础镜像
- 定期安全扫描和更新补丁
- 删除镜像中的 setuid 和 setgid 权限
  - `RUN find / -perm +6000-type f-exec chmod a-s {} \;|| true`
- 配置Docker守护程序的TLS身份验证
- 如非必要 禁止容器间通信
- rootless Docker
  - https://get.docker.com/rootless
- 使用 Seccomp 限制 syscall
- 构建环境和在线环境分开
- 证书校验

11.6.6 Docker 环境识别

1. Docker内

- MAC地址为 `02:42:ac:11:00:00` - `02:42:ac:11:ff:ff`
- `ps aux` 大部分运行的程序 pid 都很小
- `cat /proc/1/cgroup` docker的进程
- docker 环境下存在 `.dockerenv`
- 部分容器中缺少许多常用的命令如 `ping` 等

2. Docker外

- `/var/run/docker.sock` 文件存在
- `2375` / `2376` 端口开启

11.6.7 参考链接

- A House of Cards An Exploration of Security When Building Docker Containers
- Privileged Docker Containers
- 32c3 docker writeup
- 打造安全的容器云平台
- Docker security
- 容器安全
- CVE-2017-7494 Docker沙箱逃逸
- Docker容器安全性分析
- AppArmor security profiles for Docker
- Docker Bench for Security
- `Docker安全性与攻击面分析 <https://mp.weixin.qq.com/s/d9D3z13uCOJoJzplpu3WJQ>` _
- Pfleeger C P , Pfleeger S L , Theofanos M F . A methodology for penetration testing[J]. Computers & Security, 1989, 8(7):613-620.

11.7 APT

11.7.1 简介

APT (Advanced Persistent Threat)，翻译为高级持续威胁。2006年，APT攻击的概念被正式提出，用来描述从20世纪90年代末到21世纪初在美国军事和政府网络中发现的隐蔽且持续的网络攻击。

APT攻击多用于指利用互联网进行网络间谍活动，其目标大多是获取高价值的敏感情报或者控制目标系统，对目标系统有着非常严重的威胁。

发起APT攻击的通常是一个组织，其团体是一个既有能力也有意向持续而有效地进行攻击的实体。个人或者小团体发起的攻击一般不会被称为APT，因为即使其团体有意图攻击特定目标，也很少拥有先进和持久的资源来完成相应的攻击行为。

APT的攻击手段通常包括供应链攻击、社会工程学攻击、零日攻击和僵尸网络等多种方式。其基于这些攻击手段将将自定义的恶意代码放置在一台或多台计算机上执行特定的任务，并保持在较长的时间内不被发现。

和传统的大面积扫描的攻击方式不同，因为APT攻击通常只面向单一特定的目标，且多数攻击会综合一系列手段来完成一次APT攻击，使其有着非常高的隐蔽性和复杂性，让对APT攻击的检测变得相当困难。顾名思义，APT的特征主要体现在下面这三个方面。

11.7.2 高级性（Advanced）

APT攻击会结合当前所有可用的攻击手段和技术，使得攻击具有极高的隐蔽性和渗透性。

网络钓鱼就是其中的一种攻击方式。攻击者通常会结合社会工程学等手段来伪造可信度非常高的电子邮件，冒充目标信任的公司或者组织来发送难以分辨真假的对目标诱惑度很高恶意电子邮件。通过这些邮件来诱使被受害者访问攻击者控制的网站或者下载恶意代码。

APT攻击通常还会采用其他的方式来伪装自己的攻击行为，从而实现规避安全系统检测的目的。比如有的恶意代码会通过伪造合法签名来逃避杀毒软件检测。以震网病毒为例，其在攻击时就使用了白加黑的模式，利用合法的证书对其代码进行了签名，这种攻击方式会使得大部分恶意代码查杀引擎会直接认为恶意代码是合法的，而不进行任何的检测。

除了利用合法签名绕过检测，APT攻击者在攻击过程中也经常利用第三方的站点作为媒介来攻击目标，而不是使用传统的点到点攻击模式。这种模式通常被称为水坑攻击。

水坑攻击是一种入侵的手法，一般来说，是在攻击者对目标有一定了解后，确定攻击目标经常访问的网站，而后入侵其中的一个或几个网站，并对这些网站植入恶意代码，最后来实现借助该网站感染目标的能力。因为这种攻击借助了目标信任的第三方网站，攻击的成功率相对钓鱼攻击来说要高出很多。

另外一个能体现APT攻击高级性的特征是零日漏洞，目前国际上黑市一个零日漏洞的价格在数十万到数百万不等，每一个零日漏洞的稳定利用都需要大量的资源投入。而在APT攻击中，零日漏洞的利用非常广泛。以APT28为例，据统计，仅2015年一年当中APT28在攻击中就至少使用了六个零日漏洞。

11.7.3 持续性（Persistent）

和传统的基于短期利益的网络攻击有很大的不同。APT攻击的过程通常包括多个实施阶段。攻击者很很多情况下都是使用逐层渗透的方式来突破高级的防御系统，整个攻击过程一般持续时间会达到几个月甚至数年。一般来说，APT攻击可以分为以下几个阶段。

1. 侦查阶段

为了能够找到目标的脆弱点，攻击者通常会做大量的准备工作。在这个阶段攻击者多会使用基于大数据分析的隐私收集或者基于社会工程学的攻击来收集目标的信息，为了之后的攻击做出充分的准备。

2. 初次入侵阶段

基于初次侦查的信息，攻击者通常能收集到目标所使用的软件、操作系统系统版本等信息。在获取这些信息后，攻击者可以挖掘软件对应版本的零日漏洞或者使用已知漏洞来对系统做出初期的入侵行为，获取对目标一定的控制权限。

3. 权限提升阶段

在复杂网络中，攻击者初次入侵所获得的权限通常是较低的权限，而为了进一步的攻击，攻击者需要获取更高的权限来完成其需要的攻击行为。在这个阶段，攻击者通常会使用权限提升漏洞或者爆破密码等行为来实现权限提升的目的，最后获得系统甚至域的管理员权限。

4. 保持访问阶段

在成功入侵目标计算机并且有一定的权限之后，攻击者一般会使用各种方式来保持对系统的访问权限。其中一个比较常用的方式是窃取合法用户的登录凭证。当获取了用户的访问凭证之后，可以使用远程控制工具（RAT，Remote Access Tools）来建立连接，并在连接建立之后，植入特定的后门来达到持续控制的效果。

5. 横向扩展阶段

当攻击者掌握一定的目标之后，会以较慢且较隐蔽的方式逐渐在内网扩散。主要方式是先在内网进行一定的侦查。基于这些侦查，获得内网计算机的相关信息，并结合这些信息使用软件漏洞或者弱密码爆破等手段来进行横向的进一步渗透，获取更多的权限和信息。

6. 攻击收益阶段

APT攻击的主要目的是窃取目标系统的信息或对其造成一定的破坏。在完成横向扩展控制一定的内网机器后。以收集信息为目标的攻击者会使用加密通道的方式把获取的信息逐渐回传并消除入侵痕迹。而以造成破坏为目标的攻击，则在这个阶段进行相应的攻破坏。

11.7.4 威胁性（Threat）

和传统攻击不同，APT攻击的攻击手段和方案大都是针对特定的攻击对象和目的来设计。相对其他攻击，攻击者有着非常明确的目标和目的，很少会使用自动化的攻击方式，而是精确的攻击。

另外APT的目标多是政府机构、金融、能源等敏感企业、部门，一旦这些目标被成功攻击，其影响往往十分巨大。据目前已知的信息，在美国、俄罗斯等国的大选中，以及欧洲一些政治事件中，都有APT攻击出现。APT攻击已经成为国家之前斗争的一种重要手段。

11.7.8 相关事件

- 2010年伊朗震网病毒
- 2013美国棱镜门事件
- …

11.7.9 IoC

IoC (Indicators of Compromise) 在取证领域被定义为计算机安全性被破坏的证据。

常见的 IoC 有以下几种：

- hash
- IP
- 域名
- 网络
- 主机特征
- 工具
- TTPs

11.7.10 参考链接

- APT 分析及 TTPs 提取

11.8 近源渗透

11.8.1 USB攻击

1. BadUSB

通过重新编程USB设备的内部微控制器，来执行恶意操作，例如注册为键盘设备，发送特定按键进行恶意操作。

2. AutoRUN

根据主机配置的方式，一些操作系统会自动执行位于USB设备存储器上的预定文件。可以通过这种方式执行恶意软件。

3. USB Killer

通过特殊的USB设备基于电气等方式来永久销毁设备。

4. 侧信道

通过改装USB增加一些监听/测信道传输设备。

5. HID攻击

HID(human interface device)指键盘、鼠标等用于为计算机提供数据输入的人机交互设备。HID攻击指攻击者将特殊的USB设备模拟成为键盘，一旦连接上计算机就执行预定的恶意操作。HID攻击可以基于Android设备、数据线设备等实施。

11.8.2 Wi-Fi

1. 密码爆破

基于WPA2的验证方式，Wi-Fi可以通过抓握手包的方式进行线下的密码爆破。

2. 信号压制

可以使用大功率的设备捕获握手包并模仿目标AP，从而实现中间人攻击。

11.8.3 门禁

1. 电磁脉冲

部分电子门禁和电子密码锁的电子系统中集成电路对电磁脉冲比较敏感，可以通过外加电磁脉冲(Electromagnetic Pulse，EMP)的方式破坏设备，来实现打开的效果。

2. IC卡

基于变色龙等设备可以使用模拟、破解、复制IC卡破解门禁。

11.8.4 参考链接

- 近源渗透硬件指北
- 红蓝对抗之近源渗透

11.9 常见术语

11.9.1 系统相关

- WMI (Windows Management Instrumentation)

11.9.2 网络相关

1. 网络协议

- 轻型目录访问协议 (Lightweight Directory Access Protocol, LDAP)
- SMB (Server Message Block)
- SMTP (Simple Mail Transfer Protocol)
- 简单网络管理协议 (Simple Network Management Protocol, SNMP)
- POP3 (Post Office Protocol 3)
- IMAP (Internet Mail Access Protocol)
- HTTP (HyperText Transfer Protocol)
- HTTPS (HyperText Transfer Protocol over Secure Socket Layer)
- 动态主机配置协议 (Dynamic Host Configuration Protocol, DHCP)
- RPC (Remote Procedure Call)
- Java调试线协议 (Java Debug Wire Protocol, JDWP)
- NFS (Network File System)
- 服务主体名称 (Service Principal Names, SPN)

2. 路由系统

- 自治系统 (Autonomous System, AS)
- 内部网关协议 (Interior Gateway Protocol, IGP)
- 外部网关协议 (External Gateway Protocol, EGP)
- 域内路由选择 (interdomain routing)
- 域间路由选择 (intradomain routing)
- 路由信息协议 (Routing Information Protocol, RIP)
- 开放最短路径优先 (Open Shortest Path First, OSPF)
- 动态路由协议 (Dynamic Routing Protocols, DRP)
- 首跳冗余性协议 (First Hop Redundancy Protocols, FHRP)
- 热备份路由器协议 (Hot Standby Router Protocol, HSRP)
- 虚拟路由冗余协议 (Virtual Router Redundancy Protocol, VRRP)
- 网关负载均衡协议 (Gateway Load Balancing Protocol, GLBP)
- 网络地址转换 (Network Address Translation, NAT)
- 点对点协议 (Point-to-Point Protocol, PPP)
- 生成树协议 (Spanning Tree Protocol, STP)

3. 网络应用

- 证书透明度 (Certificate Transparency, CT)
- DNS证书颁发机构授权 (DNS Certification Authority Authorization, CAA)
- 应用级网关 (Application Level Gateway, ALG)

11.9.3 开发相关

- REST (Representation State Transformation)
- 持续集成 (Continuous Integration, CI)
- 持续交付 (Continuous Deployment, CD)
- 函数即服务 (Function as a Service, FaaS)
- 容器即服务 (Container as a Service, CaaS)
- 软件即服务 (Software as a Service, SaaS)
- 平台即服务 (Platform as a Service, PaaS)
- 基础设施即服务 (Insfrastructure as a Service, IaaS)

11.9.4 安全相关

- 缺点 (defect / mistake)
  - 软件在实现上和设计上的弱点
  - 缺点是缺陷和瑕疵的统称
- 缺陷 (bug)
  - 实现层面的软件缺点
  - 容易被发现和修复
  - 例如：缓冲区溢出
- 瑕疵 (flaw)
  - 一种设计上的缺点，难以察觉
  - 瑕疵往往需要人工分析才能发现
  - 软件系统中错误处理或恢复模块，导致程序不安全或失效
- 漏洞 (vulnerability)
  - 可以用于违反安全策略的缺陷或瑕疵
- IAST (Interactive Application Security Testing)
- DAST (Dynamic Application Security Testing)
- SAST (Static Application Security Testing)
- ATT&CK™ (Adversarial Tactics, Techniques, and Common Knowledge, ATT&CK)

1. 安全开发

- 安全信息和事件管理 (Security Information Event Management, SIEM)
- 自动化响应SOAR模型 (Security Orchestration, Automation and Response, SOAR)
- SDL (Security Development Lifecycle)

2. 安全策略

- 跨域资源共享策略 (Cross-Origin Resource Sharing, CORS)
- 发件人策略框架 (Sender Policy Framework, SPF)
- 域名密钥识别邮件 (DomainKeys Identified Mail, DKIM)
- 基于域名的消息认证报告与一致性协议 (Domain-based Message Authentication, Reporting and Conformance, DMARC)
- DNSSEC (The Domain Name System Security Extensions)
- 基于DNS的命名实体身份验证 (DNS-based Authentication of Named Entities, DANE)

11.9.5 攻击相关

1. 漏洞类型

- 跨站脚本攻击 (Cross Site Scripting, XSS)
- 跨站请求伪造 (Cross-Site Request Forgery, CSRF)
- 中间人攻击 (Man-in-the-middle, MITM)
- 服务端请求伪造 (Server Side Request Forgery, SSRF)
- 高级持续威胁 (Advanced Persistent Threat, APT)

2. 攻击方式

- 鱼叉攻击 (Spear Phishing)
- 水坑攻击 (Water Holing)

11.9.6 防御相关

- IoC (Indicators of Compromise)

1. 防御技术

- 网络检测响应 (Network-based Detection and Response, NDR)
- 终端检测响应 (Endpoint Detection and Response, EDR)
- 托管检测响应 (Managed Detection and Response, MDR)
- 扩展检测响应 (Extended Detection and Response, XDR)
- 自适应安全架构 (Adaptive Security Architecture, ASA)
- 零信任网络访问 (Zero Trust Network Access, ZTNA)
- 云安全配置管理 (Cloud Security Posture Management, CSPM)

2. 防护设施

- 入侵检测系统 (Intrusion Detection System, IDS)
- 主机型入侵检测系统 (Host-based Intrusion Detection System, HIDS)
- RASP (Runtime Application Self-protection)
- 统一端点管理 (Unified Endpoint Management, UEM)

11.9.7 运维

- 智能运维 (Artificial Intelligence for IT Operations, AIOps)
- 风险和脆弱性评估 (Risk and Vulnerability Assessments, RVA)

11.9.8 认证

- 双因素认证 (Two-Factor Authentication, 2FA)
- 多因素认证 (Multi-Factor Authentication, MFA)
- 一次性密码 (One-Time Password, OTP)