


浙江大学电气工程学院
College of Electrical Engineering Zhejiang University

网络安全导论

软件与通讯安全

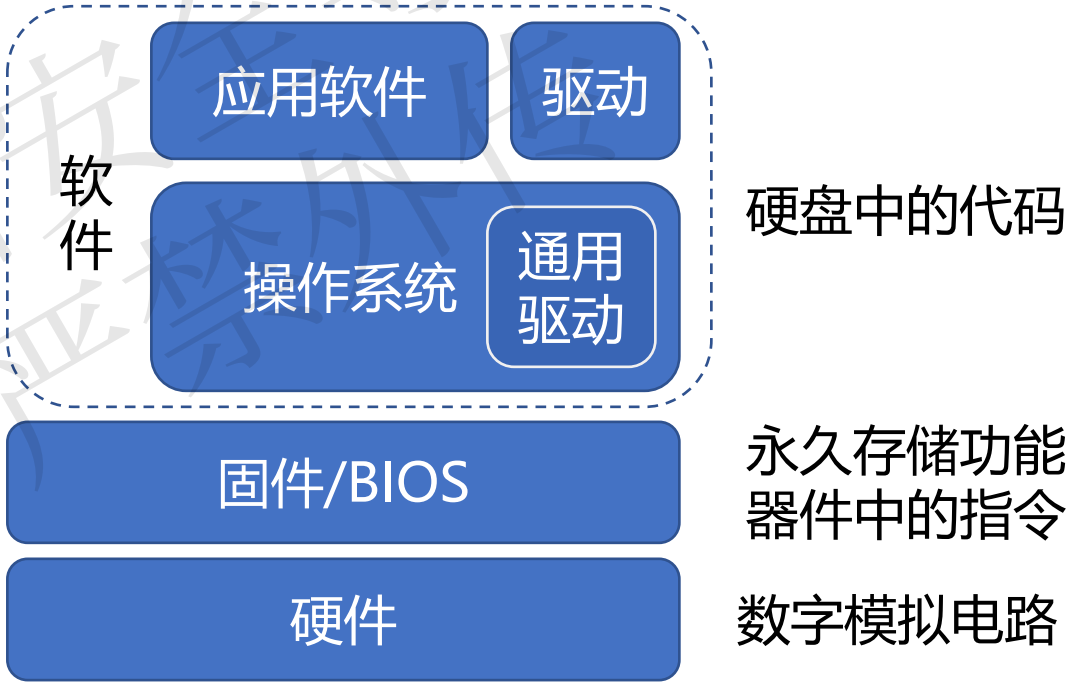
1. 概述、基础知识
2. 加密与认证技术
3. **软件与通讯安全**
4. 电力工控系统安全
5. 物联网终端安全
6. 智能无人系统安全

3.1

软件概念、固件结构

计算机结构

- 在互联网中，计算机可以分为软件、固件、硬件三个部分。其中硬件是由数字模拟电路构成，而固件和软件都是由指令构成，以代码的形式呈现。



计算机结构

固件的起源、定义

- Ascher Opler在1967年的Datamation文章中第一次提到术语**firmware**。它最初是指存储在可写入控制存储器（一个小型专用高速存储器）的内容，包含定义和实现了计算机指令集的微代码，这些微代码可以加载或修改中央处理单元（CPU）执行的指令。最初，固件与硬件（CPU本身）和软件（在CPU上执行的正常指令）都不同，它不是由CPU机器指令组成的，而是由机器指令实现中涉及的较低级别的微代码组成的。它存在于硬件和软件之间的边界上，因此名称为firmware。

固件的起源、定义

- 固件（Firmware）是写入EROM（可擦写只读存储器）或EEPROM(电可擦可编程只读存储器)等具有永久存储功能器件中的二进制程序。它是一个系统最为基础也最为底层的工作软件。

固件结构

- 固件头
- 引导程序部分（BootLoader）
- 可压缩内核部分
- 可压缩根文件系统部分
- 其它应用文件

```
DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             TP-Link firmware header, firmware ver
: 0x0, product version: 121634823, kernel load address: 0x0, kernel
4, kernel length: 512, rootfs offset: 849197, rootfs length: 10485
h: 0
13424         0x3470          U-Boot version string, "U-Boot 1.1.4
13472         0x34A0          CRC32 polynomial table, big endian
14784         0x39C0          uImage header, header size: 64 bytes,
:50:33, image size: 35910 bytes, Data Address: 0x80010000, Entry Po
x, CPU: MIPS, image type: Firmware Image, compression type: lzma, i
14848         0x3A00          LZMA compressed data, properties: 0x5
d size: 93912 bytes
131584        0x20200         TP-Link firmware header, firmware ver
, product version: 121634823, kernel load address: 0x0, kernel entr
rnel length: 512, rootfs offset: 849197, rootfs length: 1048576, bo
132096        0x20400         LZMA compressed data, properties: 0x5
d size: 2495224 bytes
1180160       0x120200        Squashfs filesystem, little endian, v
es, 622 inodes, blocksize: 131072 bytes, created: 2016-07-08 11:00:
```

固件结构

固件头

嵌入式设备固件是.bin格式二进制镜像文件，是经过编译之后压缩得到的，其内部包含着大量的二进制流数据，固件头的数据信息始终是处在二进制数据流的初始部分，以偏移量的不同组合方式表示着嵌入式设备**固件的相关特征信息**并占据一定的起始位。特征字段组成不同的组合表示成该固件设备类型的处理器平台架构、内核版本、根文件系统格式等特征字段信息。

```
DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             TP-Link firmware header, firmware version: 0.-15290.3, image version: "", product ID
: 0x0, product version: 121634823, kernel load address: 0x0, kernel entry point: 0x80002000, kernel offset: 406374
4, kernel length: 512, rootfs offset: 849197, rootfs length: 1048576, bootloader offset: 2883584, bootloader lengt
h: 0
```

固件结构

引导程序部分（BootLoader）

在嵌入式操作系统中，BootLoader是在操作系统内核运行之前运行。可以初始化硬件设备、建立内存空间映射图，从而将系统的软硬件环境带到一个合适状态，以便为最终调用操作系统内核准备好正确的环境。在嵌入式系统中，通常并没有像BIOS那样的固件程序（注，有的嵌入式CPU也会内嵌一段短小的启动程序），因此整个系统的加载启动任务就完全由BootLoader来完成。在一个基于ARM7TDMI core的嵌入式系统中，系统在上电或复位时通常都从地址0x00000000处开始执行，而在这个地址处安排的通常就是系统的BootLoader程序。常见的引导程序包括：U-Boot、RedBoot、Blob等。

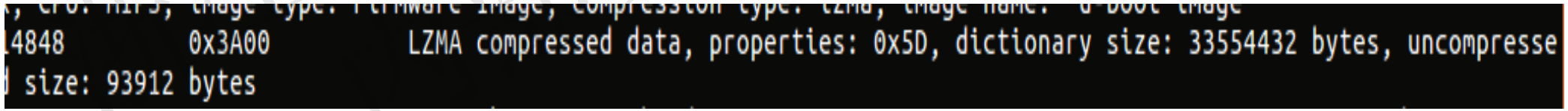
```
h: 0
13424         0x3470          U-Boot version string, "U-Boot 1.1.4 (Jul  8 2016 - 18:50:33)"
13472         0x34A0          CRC32 polynomial table, big endian
```

固件结构

可压缩内核部分，可压缩根文件系统部分

内核尺寸伸缩性强，能够适应不同配置的硬件平台。比如，一个极端的情况下，内核尺寸必须维持在10K以内，以支撑内存和CPU性能都很受限的嵌入式设备如传感器，这时候内核具备基本的任务调度和通信功能即可。在另外一个极端的情况下，内核必须具备完善的线程调度、内存管理、本地存储、复杂的网络协议、图形用户界面等功能，以满足高配置的智能物联网终端的要求。

绝大多数嵌入式设备固件中的操作系统内核支持多样文件系统类型，但是也有少数设备固件中的操作系统只支持某些特定文件系统类型。



固件结构

其他应用程序

通常利用Bin walk能够将固件解包，同时看到之前压缩的文件系统，文件系统中会存在嵌入式设备实际运行的一些应用程序。

```
ly@ubuntu:~/Desktop/gujian/_RT-AC54U.trx.extracted/squashfs-root$ ls
bin  dev  etc  etc_ro  home  init  lib  media  mnt  opt  proc  sbin  sys  tmp  usr  var  www
ly@ubuntu:~/Desktop/gujian/_RT-AC54U.trx.extracted/squashfs-root$ cd www/
ly@ubuntu:~/Desktop/gujian/_RT-AC54U.trx.extracted/squashfs-root/www$ ls
Advanced_ACL2g_Content.asp          Advanced_Extensions_SS_list.asp       Advanced_WAN_Content.asp
Advanced_ACL_Content.asp            Advanced_Extensions_SSR_Server.asp    Advanced_WGuest2g_Content.asp
Advanced_APLAN_Content.asp          Advanced_Extensions_SS_Server.asp     Advanced_WGuest_Content.asp
Advanced_BasicFirewall_Content.asp  Advanced_Extensions_wifidog.asp       Advanced_Wireless2g_Content.asp
Advanced_Console_Content.asp        Advanced_Firewall_Content.asp         Advanced_Wireless_Content.asp
```

固件主要安全风险

- 对于用户
- 对于厂商

固件主要安全风险

对于用户

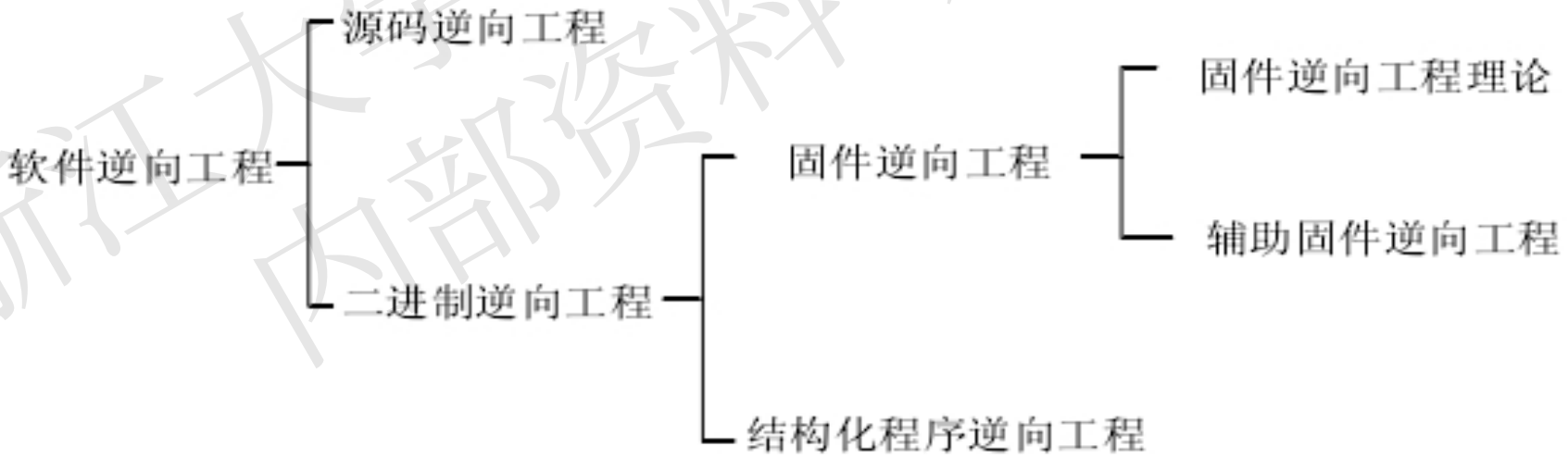
固件的安全问题主要是固件本身代码安全缺陷。这类安全问题主要包括**固件漏洞**、**固件恶意代码**以及**固件后门**。

- **固件漏洞**主要是指固件厂商在开发固件的时候，犯了一些编程上的错误。
- **固件恶意代码**则是指恶意人员在设备出厂前或者通过入侵之后在固件中注入的具有非法目的的代码程序。
- **固件后门**本质上并不是代码缺陷，他们一般是那些绕过安全性控制而获取对程序或系统访问权的程序方法。

固件主要安全风险

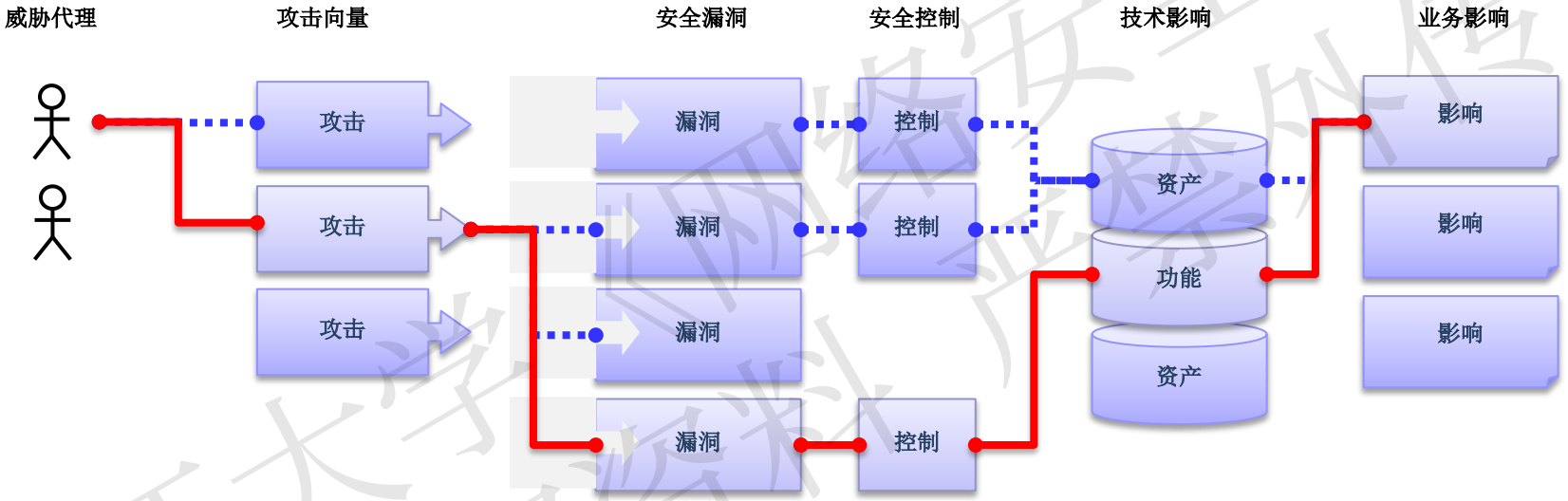
对于厂商

由于一款好的物联网设备能够带来巨大的经济效益，攻击者能够通过逆向工程来获得固件的设计信息，侵害生产厂商的知识产权。除此之外，攻击者还可以根据固件的设计信息对固件进行漏洞的挖掘和恶意代码的注入，从而影响用户安全。这些恶意的逆向攻击在造成巨大经济损失的同时，侵害了软件作者的成果，严重损害物联网的健康发展。



软件安全风险

- 攻击者可以通过软件程序中许多不同的路径方法去危害你的业务或者企业组织。每种路径方法都代表了一种风险，这些风险可能会，也有可能不会严重到值得去关注。



- 有时，这些路径方法很容易被发现并被利用，但有的则非常困难。同样，所造成危害的范围也从没有危害，到有可能完全损害你的整个业务。为了确定你的企业的风险，你可以结合其产生的技术影响和对企业的业务影响，去评估威胁代理、攻击向量和安全漏洞的可能性。总之，这些因素决定了全部的风险。

3.2 不安全的web接口/不安全的认证与授权/不安全的系统和程序

软件漏洞概述

软件漏洞概述

- 不安全的Web接口
  - SQL注入
  - XSS攻击
- 不安全的认证和授权
  - 弱口令登录
- 不安全的系统和程序
  - 缓冲区溢出漏洞
  - 格式化字符串漏洞
  - 空指针引用漏洞
  - 释放后重用漏洞
  - 条件竞争漏洞

不安全的Web接口

SQL注入攻击

SQL注入是发生于应用程序与数据库层的安全漏洞。简而言之，是在输入的字符串之中注入SQL指令，在设计不良的程序当中忽略了字符检查，那么这些注入进去的恶意指令就会被数据库服务器误认为是正常的SQL指令而运行，因此遭到破坏或是入侵。

攻击案例

应用程序在下面存在漏洞的SQL语句的构造中使用不可信数据：

```
String query = "SELECT * FROM accounts WHERE
custID='" + request.getParameter("id") +"'";
```

攻击者在浏览器中将“id”参数的值修改成 ' or'1'='1. 这样查询语句的意义就变成了从帐户数据库中返回所有的记录，而不是只有目标客户的信息。

```
http://example.com/app/accountView?id=' or '1'='1
```

在最严重的情况下，攻击者能够使用这一漏洞调用数据库中特殊的储存过程，从而达到完全接管数据库，甚至运行数据库的主机。

不安全的Web接口

XSS跨站脚本攻击

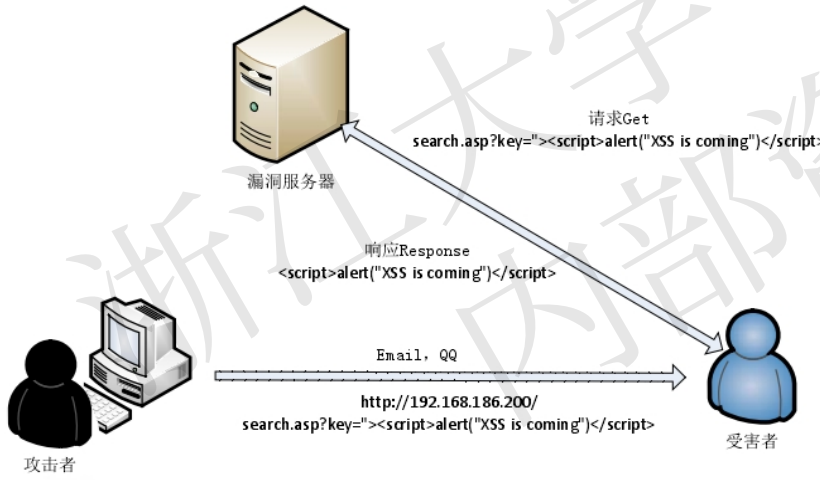
- 攻击者可以在一个Web站点越过安全边界线使用某些方式把其恶意代码注入到另一个存在漏洞的Web站点中。这些被注入的恶意代码就被认为是目标站点的代码，在受害者的浏览器中被执行时，攻击者轻松的窃取到相应的敏感数据，或者强迫用户做一些并非用户本意的事情。跨站脚本XSS漏洞的攻击具有隐蔽性、攻击也容易发起等特点，同时还具有快速扩散能力。
- 根据其特性和利用手法的不同可以将跨站脚本XSS漏洞分成2个大类型，一种是反射型跨站脚本，另一种是持久型跨站脚本。

不安全的Web接口

XSS跨站脚本攻击

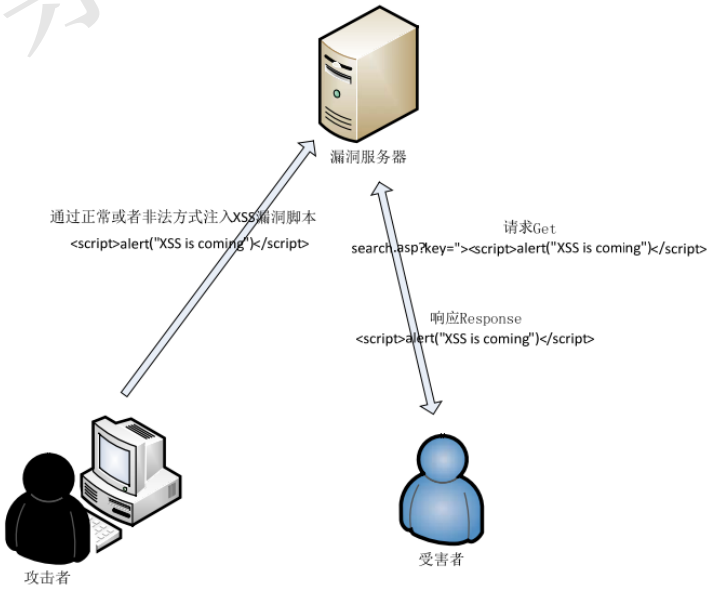
反射型XSS

- 也称非持久型XSS，参数型跨站脚本，是最常见使用最广的XSS。



持久型XSS

- 也称存储型 XSS，比反射型跨站脚本更具有威胁性，并且可能影响到 Web 服务器自身的安全。



不安全的认证和授权

弱口令

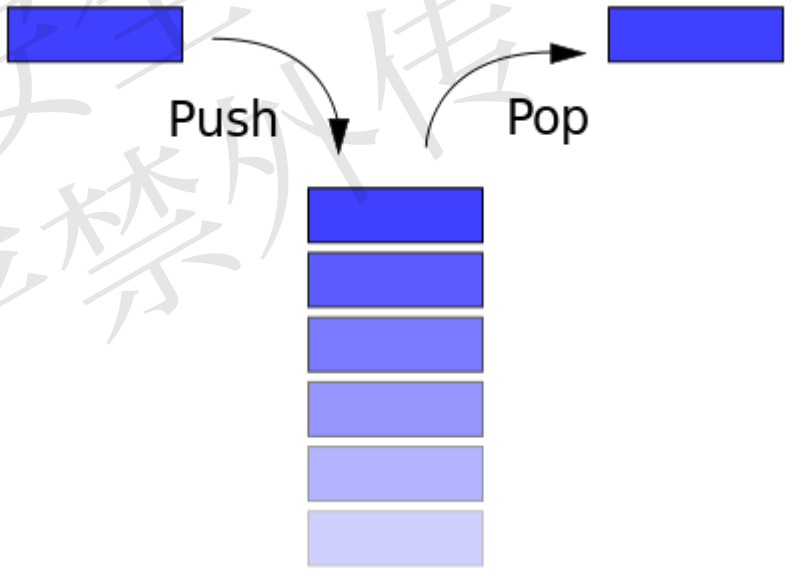
弱口令(weak password) 没有严格和准确的定义，通常认为容易被别人（他们有可能对你很了解）猜测到或被破解工具破解的口令均为弱口令。弱口令指的是仅包含简单数字和字母的口令，例如“123”、“abc”等，因为这样的口令很容易被别人破解，物联网设备遭到威胁。但是由于人们对密码记忆能力有限，而且缺乏一定的安全意识，导致弱口令在物联网设备中依旧十分普遍。

不安全的系统和程序

缓冲区溢出漏洞

- 缓冲区溢出是针对程序设计缺陷，向程序输入缓冲区写入使之溢出的内容（通常是超过缓冲区能保存的最大数据量的数据），从而破坏程序运行、趁著中断之际并获取程序乃至系统的控制权。

- 一般常见的缓冲区就是栈缓冲区和堆缓冲区。



不安全的系统和程序

缓冲区溢出漏洞

- 栈溢出是缓冲区溢出漏洞的一种。
- 通常是由于对输入数据的长度没有进行检查导致的。
- 利用栈溢出漏洞可以破坏栈上其他变量，也可以覆盖返回地址劫持程序控制流。

```
int main(int argc, char** argv)
{
    int modified;
    char buffer[8];
    modified = "0";

    gets(buffer);          // 引发缓冲区溢出
    if (modified == "1")
    {
        printf("Congratulations, you pwned it.\n");
    }
    else
    {
        printf("Please try again.\n");
    }
    return 0;
}
```

不安全的系统和程序

格式化字符串漏洞

- 格式化字符串函数是带有可变数量的参数，并将第一个参数作为格式化字符串，根据该参数来解析之后的参数。当格式化串可控时，就产生了格式化字符串漏洞。

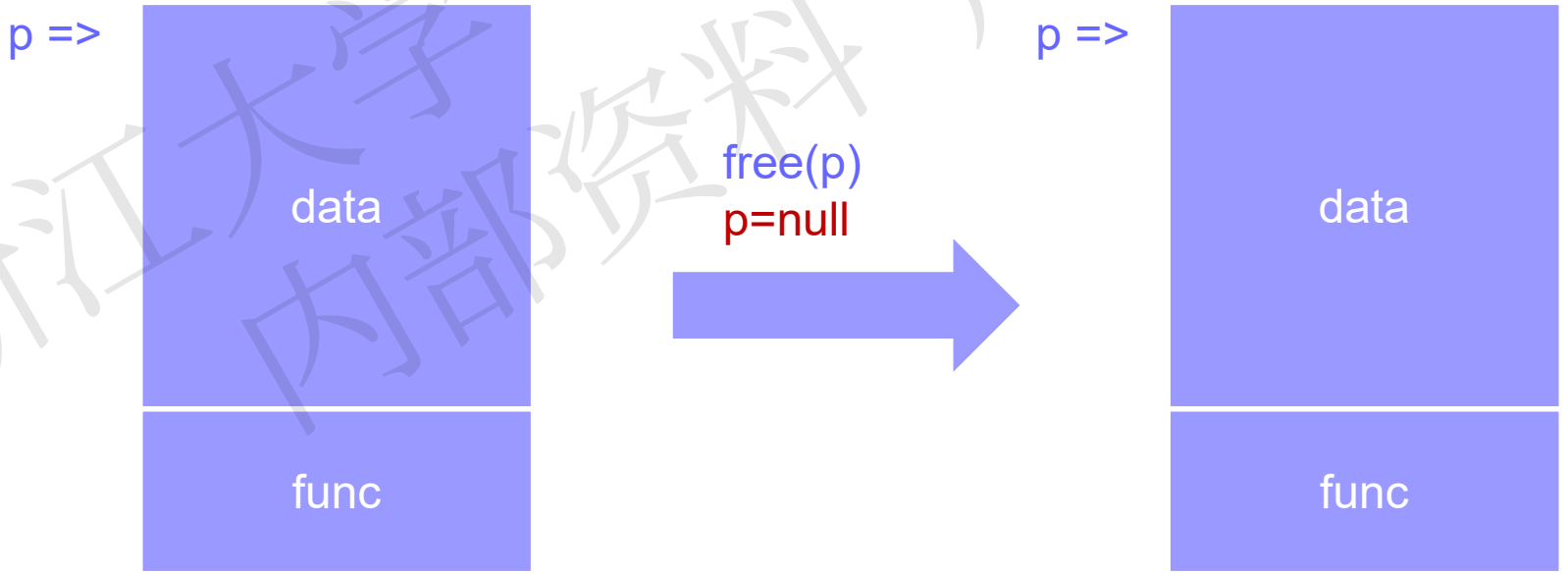
- 如我们最常见到的printf函数，

```
printf("<格式化字符串>", <参量表>);
```

不安全的系统和程序

释放后重用

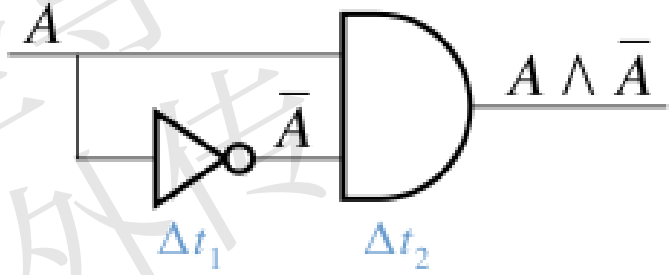
- 释放后重用是指在释放了某一块内存后，仍然对该内存进行访问，一般是由于在释放内存后没有将指向该内存区域的指针全部清空导致。一般称被释放后没有被置为null的内存指针为悬垂指针（dangling pointer）。



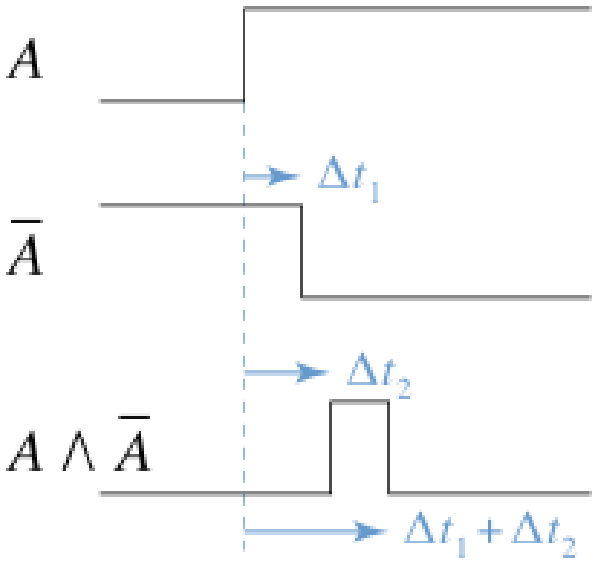
不安全的系统和程序

条件竞争漏洞

- 条件竞争是指一个系统的运行结果依赖于不受控制的事件的先后顺序，最初来自于逻辑电路中两个电信号的互相竞争。



- 目前的操作系统中大量采用并发编程，经常对资源进行共享，往往会产生条件竞争漏洞。在多线程环境下，当一个软件的运行结果依赖于进程或者线程的顺序时，就可能会出现条件竞争。



主观题 1分

设置

右边这段程序，输出应该是什么？

```
int i = 1;
void *mythread1()
{
  if(i == 1){
    sleep(3);
    if(i == 2)
      printf("hack it!\n");
    else
      printf("you can try again!\n");
  }
}

void *mythread2()
{
    sleep(1);
    i=2;
}

int main(int argc, const char *argv[])
{
    pthread_t id1,id2;
    pthread_create(&id1, NULL, (void *)mythread1,NULL);
    pthread_create(&id2, NULL, (void *)mythread2,NULL);
    pthread_join(id1,NULL);
    pthread_join(id2,NULL);
    return 0;
}
```

作答

不安全的系统和程序

条件竞争漏洞

- 条件竞争造成的影响也是多样的，轻则程序异常执行，重则程序崩溃。如果被攻击者利用，则可能会使攻击者获得相应系统的特权。

- 一个典型的例子如图所示，正常应该下应该输出”you can try again!“，因为i初始值是为1的，但是实际运行后（可以自行尝试）输出的确是”hack it!“。

```c
int i = 1;
void *mythread1()
{
  if(i == 1){
    sleep(3);
    if(i == 2)
      printf("hack it!\n");
    else
      printf("you can try again!\n");
  }
}

void *mythread2()
{
    sleep(1);
    i=2;
}

int main(int argc, const char *argv[])
{
    pthread_t id1,id2;
    pthread_create(&id1, NULL, (void *)mythread1,NULL);
    pthread_create(&id2, NULL, (void *)mythread2,NULL);
    pthread_join(id1,NULL);
    pthread_join(id2,NULL);
    return 0;
}
```

不安全的系统和程序

条件竞争漏洞

- 这里存在条件竞争的原因在于：
- 程序首先执行了 mythread1，然后在判断i等于1后执行了sleep(3)函数，也即进入了睡眠状态，而这时因为效率原因，程序不会等待mythread1，转而去执行mythread2；
- 在mythread2中，也执行了sleep(1)，但由于时间较短，mythread2可能比mythread1先“醒”过来，然后对i进行了修改；
- 之后mythread1继续执行时，i的值已经被修改为2了，于是输出”hack it! “。

```c
int i = 1;
void *mythread1()
{
  if(i == 1){
    sleep(3);
    if(i == 2)
      printf("hack it!\n");
    else
      printf("you can try again!\n");
  }
}

void *mythread2()
{
    sleep(1);
    i=2;
}

int main(int argc, const char *argv[])
{
    pthread_t id1,id2;
    pthread_create(&id1, NULL, (void *)mythread1,NULL);
    pthread_create(&id2, NULL, (void *)mythread2,NULL);
    pthread_join(id1,NULL);
    pthread_join(id2,NULL);
    return 0;
}
```

3.3 程序保护机制/逆向防护机制/软件缺陷防护

软件安全防护

软件安全防护

- 程序保护机制
  - CANNARY
  - FORTIFY
  - NX/DEP
  - PIE/ASLR
  - RELRO
- 逆向防护机制
- 软件缺陷防护

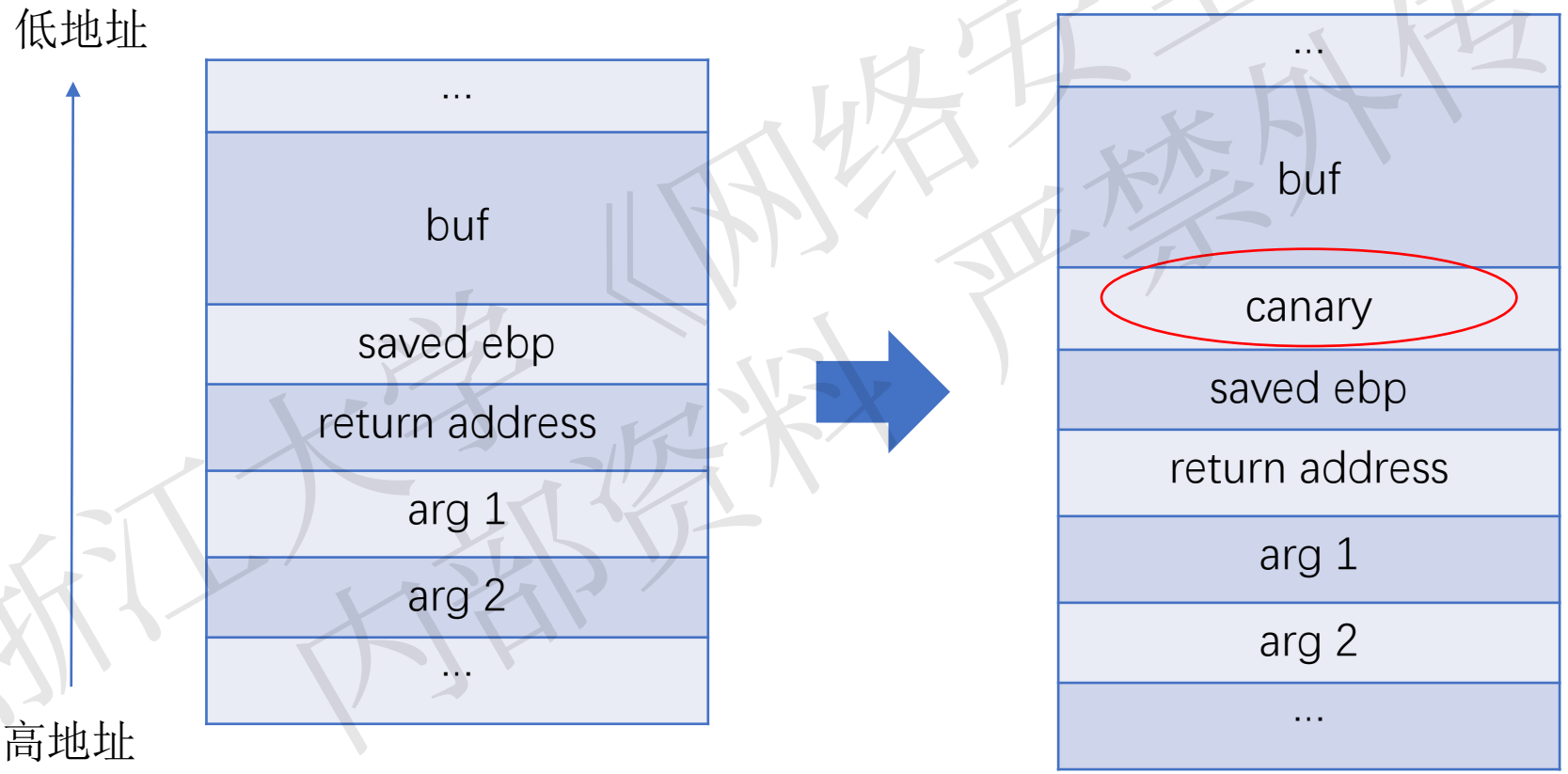
程序保护机制

- CANNARY

又称栈溢出保护，是一种缓冲区溢出攻击缓解手段。当函数存在缓冲区溢出漏洞时，攻击者可以覆盖栈上的返回地址来劫持控制流。当启用栈保护后，函数开始执行的时候会先往栈里插入cookie信息，当函数真正返回的时候会验证cookie信息是否合法，如果不合法就停止程序运行。攻击者在覆盖返回地址的时候往往也会将cookie信息给覆盖掉，导致栈保护检查失败而阻止栈溢出攻击。在Linux中我们将cookie信息称为cannary。

程序保护机制

- CANNARY



程序保护机制

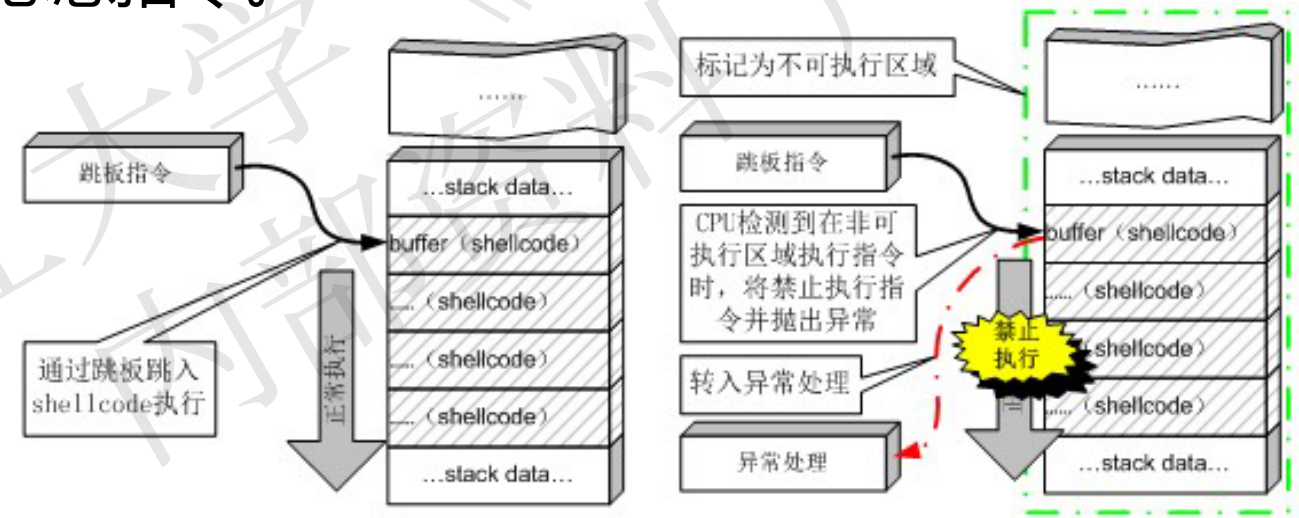
- FORNIFY

- 用于检查是否存在缓冲区溢出的错误。适用情形是程序采用大量的字符串或者内存操作函数，如memcpy，memset，stpcpy，strcpy，strncpy，strcat，strncat，sprintf，snprintf，vsprintf，vsnprintf，gets以及宽字符的变体。
- gcc -D_FORTIFY_SOURCE=1 仅在编译时进行检查
- gcc -D_FORTIFY_SOURCE=2 程序执行时也会有检查 (如果检查到缓冲区溢出，就终止程序)

程序保护机制

- NX/DEP

- NX即No-eXecute，不可执行；DEP，即数据执行保护，两者原理相同。
- 其基本原理是将数据所在内存页标识为不可执行，当程序溢出成功转入shellcode时，程序会尝试在数据页面上执行指令，此时CPU就会抛出异常，而不是去执行恶意指令。



经典溢出流程　　启用DEP后流程

程序保护机制

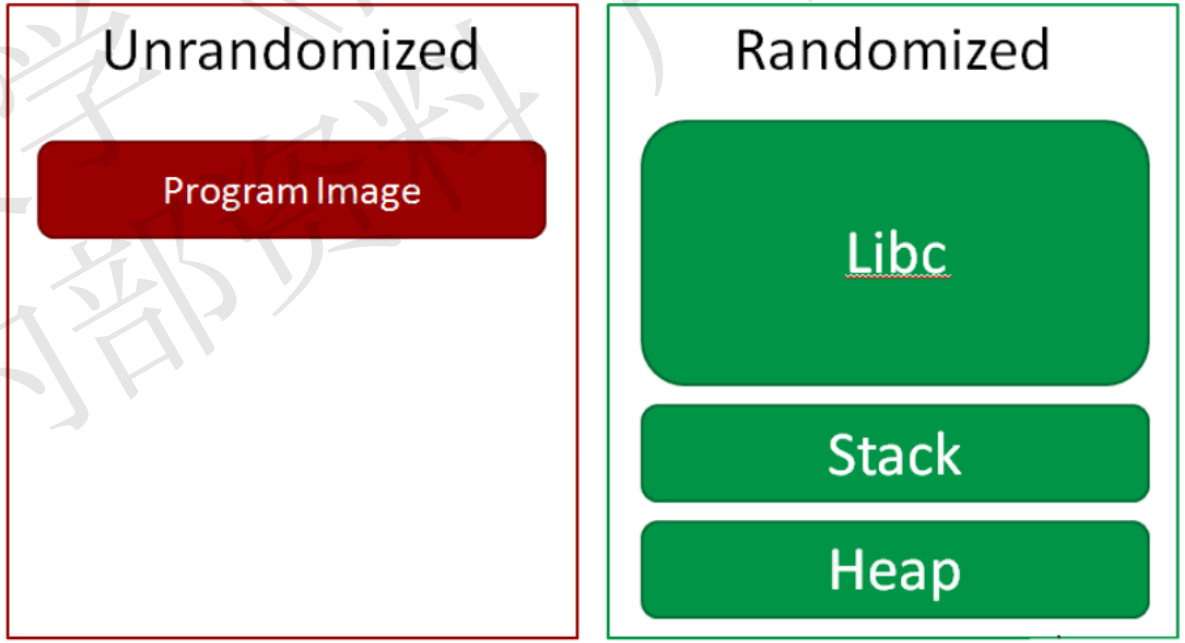
- PIE

- Position-Independent Executables，位置独立可执行。
- 程序的一个编译选项，加了PIE选项编译后程序会被当做so文件来载入
- 开启该保护后，每次执行程序时，程序基址会被映射到不同的地址上，这使得在利用缓冲区溢出和利用操作系统中存在的其他内存崩溃缺陷时采用返回导向编程方法变得较难利用。

程序保护机制

- ASLR

- 地址空间随机化
- Linux下默认开启的一种系统保护机制
- 三种级别：0不开启，1普通，2加强



程序保护机制

- RELRO

- Relocation Read Only，重定向只读保护。
- 大概实现就是由linker指定binary的一块经过dynamic linker处理过 relocation 之后的区域为只读。
- 设置符号重定向表格为只读或在程序启动时就解析并绑定所有动态符号，从而减少对GOT表（Global Offset Table，全局偏移表）的攻击。

逆向防护技术

- 世界上没有破解不了的软件，只有不值得破解的软件。换言之，只有软件的破解成本超过Hacker收益，软件才是相对安全的。逆向防护技术就是为了保护自己开发的软件不被逆向破解。
- 目前常见的软件反逆向技术有**指令加密机制**、**访问认证机制**、反静态反汇编技术、反动态反汇编技术、**代码混淆技术**以及代码自修改技术等。

4.1

通讯协议定义与分类

通讯协议定义

- 通讯协议（英语：Communications Protocol，也称传输协议）在电信中是指在任何物理介质中允许两个或多个在传输系统中的终端之间传播信息的**系统标准**，也是指计算机通信或网络设备的**共同语言**。
- 三要素：语法、语义、定时规则

语法	语义	定时规则
数据的格式、编码和信号等级等	通信内容，包括数据内、含义以及控制信息等	通信的顺序、速率匹配和排序

通讯协议功能

- **分段**：将数据分为较小的，长度受限的数据块；
- **重组**：数据接收侧重新把数据组成消息；
- **封装**：在分段形成的数据块上增加控制信息的过程；
- **连接控制**：数据通信分为无连接和面向连接两种通信方式。在无连接的方式中，每个协议数据单元传送的过程中进行独立处理；在面向连接的方式中，两个实体之间需要建立一个逻辑关系，然后通过建立的链接进行数据单元的有序传送；
- **流量控制**：接收实体对发送实体送出的数据单元的数量或速率进行限制，以避免数据单元的丢失和网络的拥塞；

通讯协议功能

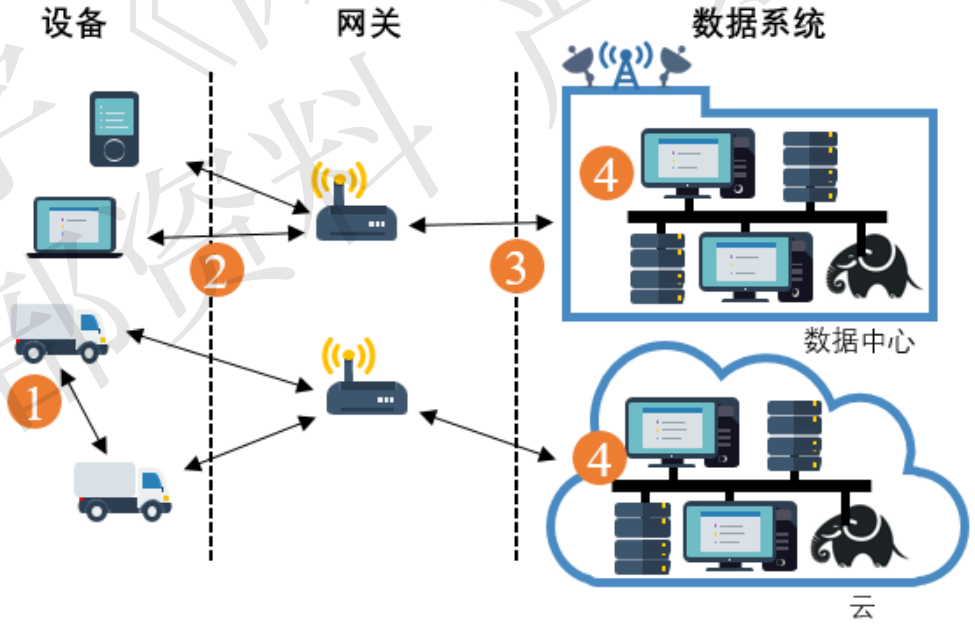
- **差错控制**：用来对协议数据单元中的数据和控制信息进行保护的。一方面，差错控制对收到的数据进行校验，在出错的情况下对整个数据单元重新进行传输，另一方面，利用定时器进行控制，当超出规定的时间没有收到确认信号则重新传输；
- **寻址**：根据消息中给出的地址信息寻找有效地址；
- **复用**：是指在一个系统上支持多个连接。例如在X.25协议中多条虚电路可以终接在一个端系统中。它可以利用端口号来实现；
- **附加功能**：协议还可以对通信实体提供各种服务，例如优先权、服务等级及安全设置等。

这些功能通过网络的各层实现，网络的每一层不一定具有上述全部功能，可以完成其中一部分功能，但不同层可以具有相同的功能。

通讯协议分类

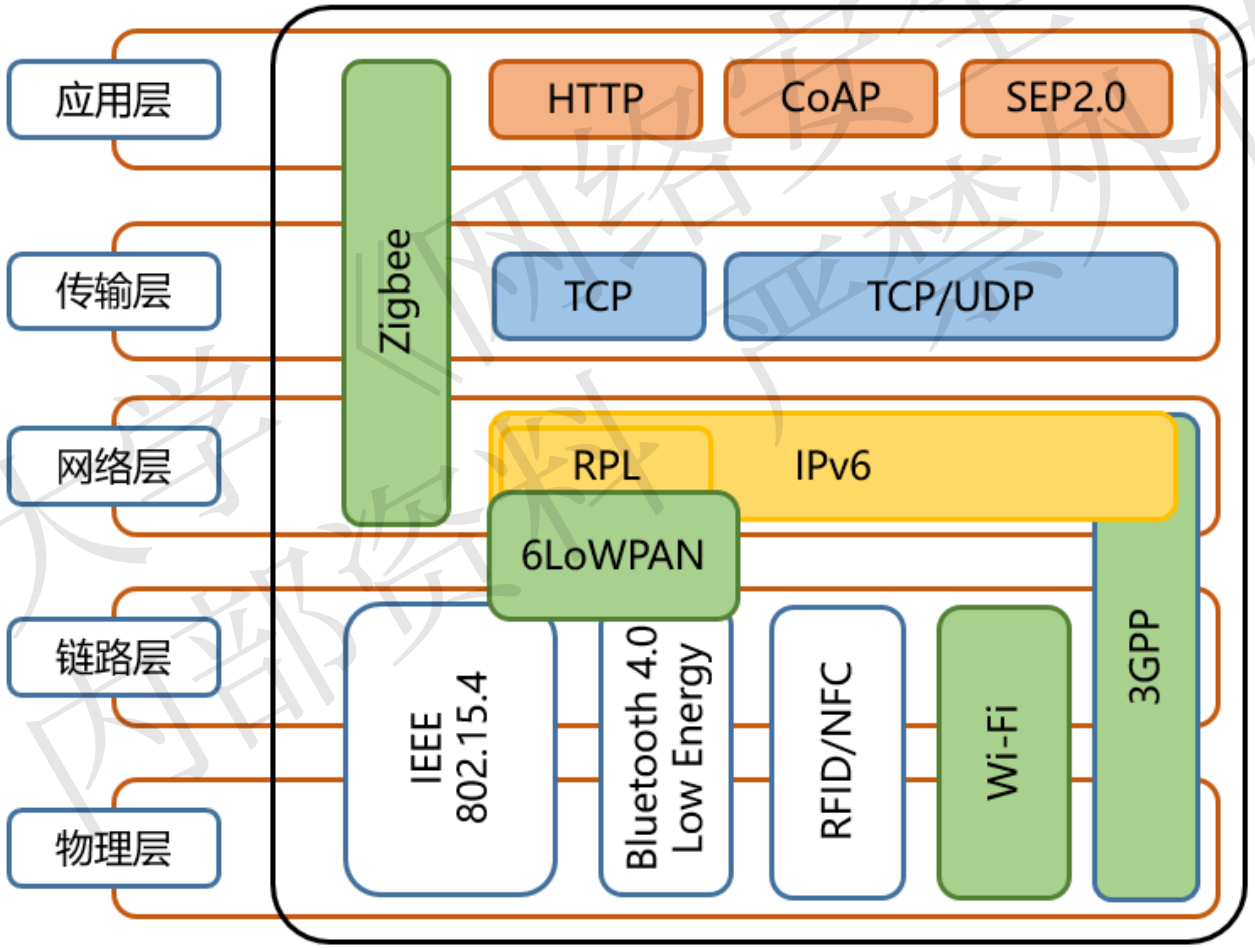
基于连接对象的通讯协议分类

(1) **端设备与端设备**之间的通讯协议；(2) **端设备与管道设备**的通讯；(3) **管道设备与云设备**之间的通讯协议；(4) **云设备与云设备**之间的通讯协议。根据连接对象属性的不同，采用的通讯协议也存在差异。



通讯协议分类

- 基于OSI模型的通讯协议分类



通讯协议分类

- 基于业务功能的通讯协议分类

分类	典型协议
基础结构类	6LowPAN, IPv4/IPv6, UDP, RPL
传输通讯类	WiFi, Bluetooth, LPWAN
消息数据类	MQTT, CoAP, AMQP, Websocket, Node
身份识别类	EPC, uCode, IPv6, URIs
设备发现类	Physical Web, mDNS, DNS-SD
设备管理类	TR-069, OMA-DM
语义语法类	JSON-LD, Web Thing Model
多层架构类	Alljoyn, IoTivity, Weave, Homekit

通讯协议分类

- 传输通讯类：定义信息**传输波段**、**数据帧结构**、**最高传输速率**等内容

协议名称	使用频率	传输速率	覆盖范围	能耗	价格
2G/3G	蜂窝频段	10Mbps	2-10公里	高	高
4G	蜂窝频段	100Mbps	1-3公里	高	高
5G	蜂窝频段	10 Gbps	100-300米	高	高
Bluetooth/BLE	2.4GHz	1,2,3Mbps	100米级别	低	低
WiFi	2.4GHz, 5GHz	0.1-54Mbps	<100米	中	低
Zigbee	2.4GHz	250kps	100米级别	低	中
Z-Wave	subGHz	40kps	30米级别	低	中
LoRa	subGHz	<50kps	1-5千米	低	中
NB-IoT	蜂窝频段	0.1-1Mbps	公里级别	中	高
802.15.4	subGHz, 2.4GHz	40,250kps	>100平方公里	低	低
SigFoX	subGHz	<1kps	公里级别	低	中
LTE Cat 0/1	蜂窝频段	1-10Mbps	公里级别	中	高
WirelessHART	2.4GHz	250kps	100米级别	中	中
Weightless	subGHz	0.1-24Mbps	公里级别	低	低

通讯协议分类

- **基础架构类：** 提供基础服务
  - **IPv4/IPv6**：用于分组交换网络互联的网络层协议，并提供跨多个IP网络的端到端数据电报传输。
  - **6LoWPAN**：IPv6低功耗无线个人局域网标准。
  - **UDP**：User Datagram Protocol，用户数据报协议。
- **消息数据类：** 应用层消息数据传输规范的协议
  - **CoAP**：CoAP协议是基于REST架构与UDP传输协议的面向资源受限设备的物联网应用层协议。
  - **MQTT**：MQTT是一种基于发布/订阅模式的“轻量级”通讯协议。
  - **XMPP**：XMPP是一种基于XML的通讯协议，用于实时应用程序，具有超强的可扩展性。

4.2

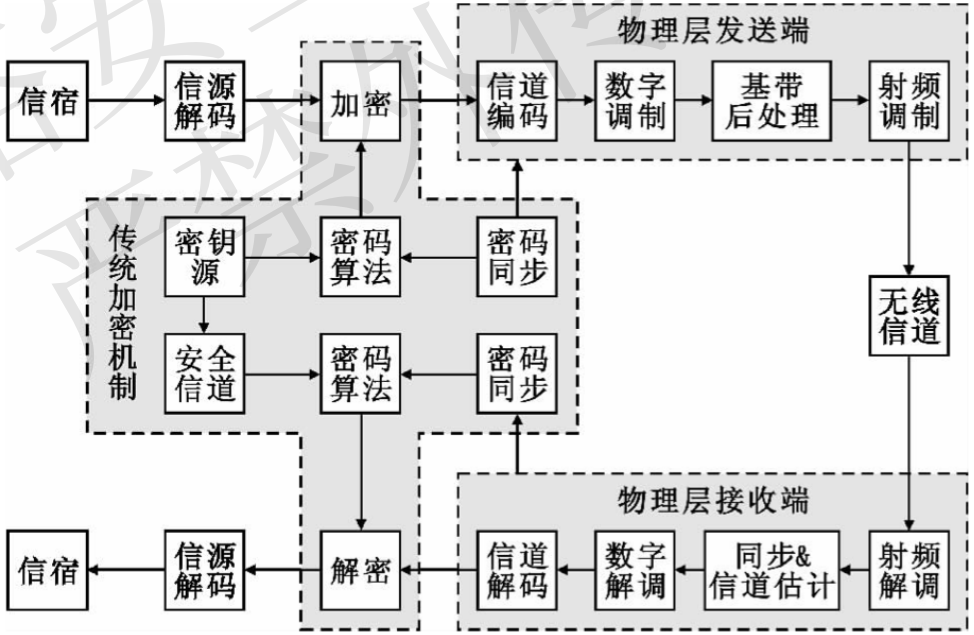
通讯安全风险

通讯安全风险

- 物理层安全风险
- MAC层安全风险
- 网络层安全风险
- 传输层安全风险
- 应用层安全风险
- 攻击后果

物理层安全风险概述

- 加密操作在物理层之外完成，加密后的密文信息输送到物理层进行无线传输。物理层的作用是将信息转换成适合无线信道传输的形式。
- 若物理层对窃听者完全透明，窃听者极易通过无线信道对传输信息进行非法接收。
- 物理层安全技术可作为在无线通信的安全框架下对传统加密机制的必要补充，对上层加密信息在无线传输过程中形成有力保护 。



采用传统加密机制的无线通信系统

物理层安全风险类型

- **窃听攻击**

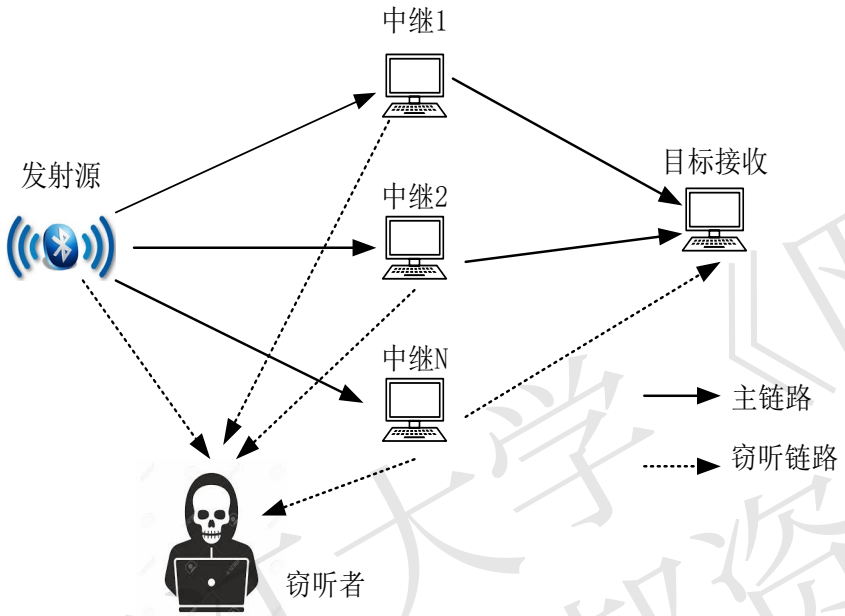
  用各种可能的合法或非法的手段窃取系统中的信息资源和敏感信息。

- **干扰攻击**

  无线网络中的恶意节点很容易产生故意干扰，破坏合法用户之间的数据通信，这被称为干扰攻击（也称为DoS攻击）。

物理层安全风险

窃听攻击



窃听攻击示意图

- 发射源通过中继的主链路发送相应的信息给目标设备；
- 窃听者通过非法手段形成所谓的窃听链路接收获取有用信息；
- 窃听者在目标设备不察觉的情况下通过相应手段对窃听信息进行处理以达成相关目的。

物理层安全风险

加密技术在窃听攻击中的应用

- 为了维护机密传输，通常采用依赖于密钥的密码技术来防止窃听攻击拦截数据传输。
- SN和DN共享一个密钥，所谓的明文首先在SN处加密，生成密文，再传输给DN。
- 密钥的加密技术能在一定程度上防止窃听攻击从而保障物理层的安全，但是针对密钥的攻击手段越来越多。
- 随着WiFi、蓝牙和ZigBee技术在物联网技术中的大量运用，物理层的信息传输存在越来越多的安全隐患。

物理层安全风险

加密技术的局限性

- **密钥攻击定义：**指的是在某些密钥传输过程中，攻击者获取了相关的密码，从而达到破解协议的目的,从而造成某些隐私信息的泄露问题。
- **例如WiFi加密安全**：密钥重新安装攻击滥用加密协议中的设计或实现缺陷，以重新安装已使用的密钥，从而达到对物理层传输信息的窃听。
- **例如ZigBee加密安全**：在密钥传输过程中，可能会以明文形式传输网络或者链接密钥，因此可能被窃取到密钥，从而解密出通讯数据，或者伪造合法设备。

物理层安全风险

- **ZigBee的三等级安全模式**
  - **非安全模式**

    默认安全模式，即不采取任何安全服务，因此可能被窃听。
  - **访问控制模式**

    通过访问控制列表(Access Control List，ACL，包含有允许接入的硬件设备MAC地址) 限制非法节点获取数据。
  - **安全模式**

    采用AES 128位加密算法进行通讯加密，同时提供有32，64，128位的完整性校验，该模式又分为标准安全模式和高级安全模式，标准模式下以明文密钥传输，高级安全模式下禁止传输密钥。

物理层安全风险

- **ZigBee使用中的安全风险**
- **窃听攻击**

  当ZigBee采用非安全模式时，对传输数据将不作加密处理，因此可能被外部窃取到通讯数据

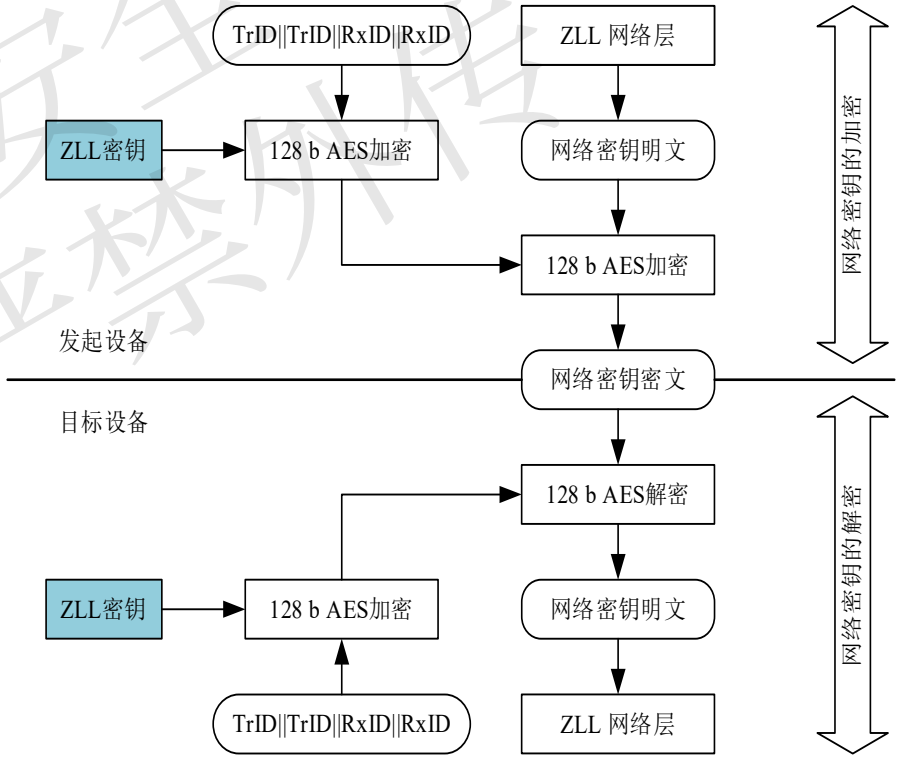
- **密钥攻击**

  由于在密钥传输过程中，可能会以明文形式传输网络/链接密钥，因此可能被窃取到密钥，从而解密出通讯数据，或者伪造合法设备。也有可能通过逆向一些智能设备固件，从中获取密钥进行通讯命令解密，然后伪造命令进行攻击。

物理层安全风险

ZigBee Light Link技术标准

- 为了解决密钥攻击问题，2012年4月，国际ZigBee联盟正式公布了ZigBee Light Link(以下简称ZLL) 技术标准。
- ZLL标准就采用了两级加密，发送端认证密钥加密设备交互过程中的识别符，产生一次性的传输密钥，传输密钥再次作为密钥加密ZLL网络密钥。接收端执行对称的解密流程恢复网络密钥。
- 此标准可以避免密钥泄露的问题，产品密钥作为商业秘密，在产品经过认证后由ZigBee联盟发布。



采用ZLL密钥的网络密钥传输机制

物理层安全风险

干扰攻击

- **定义**：干扰攻击的目的是防止授权用户访问无线网络资源，将损害合法用户的网络可用性。
- **例1**：在针对蓝牙的干扰攻击中，Bluejacking是一种滥用手机vCard功能的技术，vcard类似于名片，用户在蓝牙手机之间发送短格式的消息，接受vcard通常不需要在接收端进行交互，从而为攻击者提供了一种无需任何凭证就可以发送匿名消息的方式，这种攻击可以用移动设备上可疑的信息来恐吓用户。
- **例2**：Blueper技术，它旨在滥用某些移动设备上的蓝牙文件传输，它向目标发送大量文件传输请求，一个可能的结果是通过连续不断的弹出窗口，给用户带来了大量文件传输请求的消息。在没有用户交互或以前的身份验证的情况下，将数据写入目标设备磁盘，导致一些设备暂时停止执行或崩溃。

MAC层安全风险

MAC层概述

MAC层使多个网络节点能够借助智能信道访问控制机制访问共享介质。通常每个网络节点都配备一个NIC（网络接口控制器），并具有唯一的MAC地址，用于用户身份验证。

MAC欺骗

试图以恶意的意图改变其分配的MAC地址的攻击者被称为MAC欺骗，这是MAC层攻击的主要技术

MAC层安全风险

MAC身份盗用

MAC攻击者可能会通过分析窃听的流量来偷听网络流量并窃取合法节点的MAC地址，这被称为身份盗窃攻击。尝试身份盗用的攻击者将伪装成另一个合法的网络节点并获取对受害节点的机密信息的访问权。

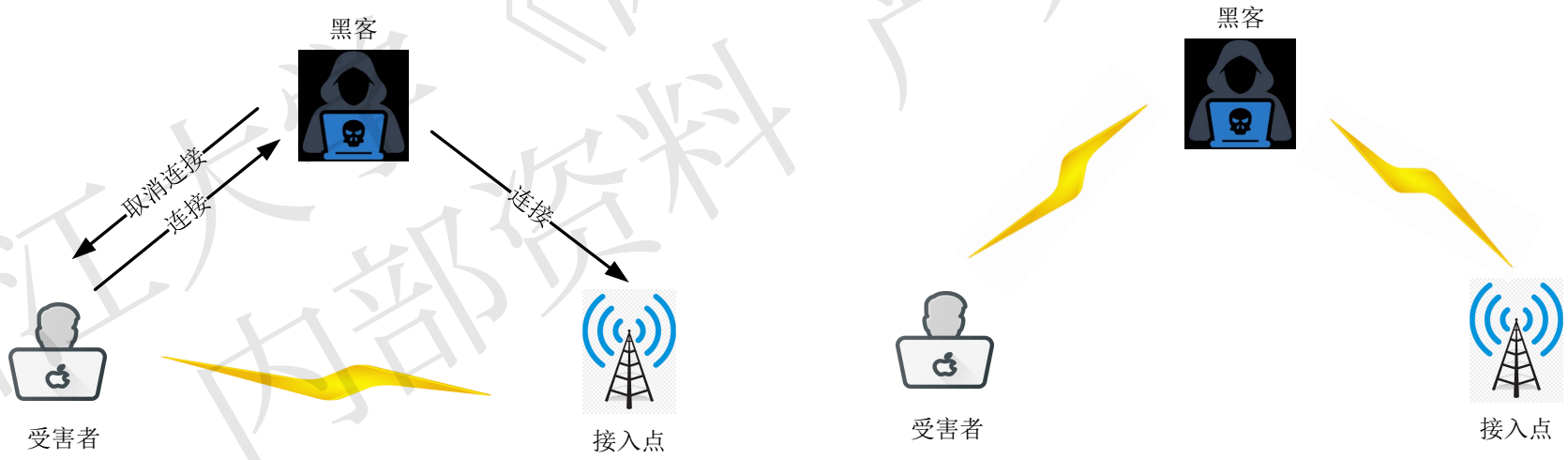
MITM（中间人）攻击

MAC层攻击还包括MITM（中间人）攻击，即攻击者与通讯的两端分别创建独立的联系，并交换其所收到的数据，使通讯的两端认为他们正在通过一个私密的连接与对方直接对话，但事实上整个会话都被攻击者完全控制。

MAC层安全风险

MITM攻击

MITM攻击是指攻击者首先“嗅探”网络的流量以拦截一对合法通信节点的MAC地址，然后冒充两个受害者并最终建立与他们的连接。通过这种方式，MITM攻击者充当受害者对之间的中继，并使他们感觉到他们通过私人连接直接相互通信。实际上，他们的会话被攻击者拦截和控制。



中间人攻击场景

网络层安全风险

网络层概述

网络层中，IP（互联网协议）被设计为主要协议，根据IP地址通过中间路由器将数据包从SN（源节点）传递到DN（目标节点）。网络层攻击的主要目的是利用IP的弱点，包括IP欺骗和劫持以及所谓的Smurf攻击。

IP欺骗与劫持

- IP欺骗用于生成伪造的IP地址，目的是隐藏攻击者的真实身份或模仿另一个网络节点以执行非法活动。
- IP劫持是劫持者为了接管另一个合法用户的IP地址而发起的另一项非法活动。如果攻击者成功劫持了IP地址，它将能够断开合法用户并通过模仿合法用户创建与网络的新连接，从而获得对机密信息的访问权限。

MAC层安全风险

- **Smurf 攻击**

  Smurf攻击是网络层中的DoS攻击，它使用IP广播地址向受害节点或受害者群体发送大量ICMP（互联网控制消息协议）数据包（带有欺骗性源IP地址）。收到ICMP请求后，受害者需要发回ICMP响应，从而导致受害者网络中出现大量流量。

MAC层安全风险

暴力攻击

背景

在网络层中，WEP和WPA是常用的网络层认证协议，而这样的认证协议常常受到攻击者的攻击。

暴力破解

在密码学中，暴力攻击是指攻击者提交许多密码或密码短语，希望最终能够正确猜测密钥，攻击者系统地检查所有可能的密码，直到找到正确的密码为止。

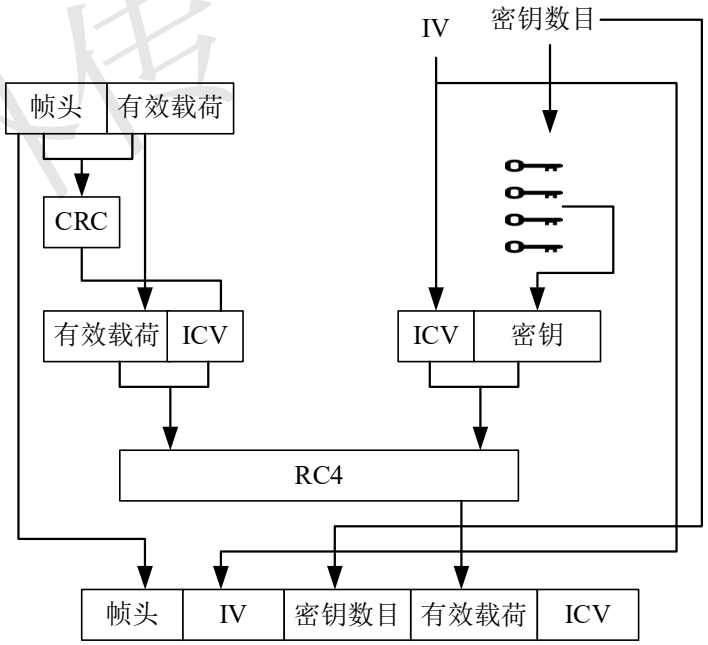
WEP加密介绍

在无线安全中，WEP曾经是无线网的标准加密算法，而如今WEP已经被WPA/WPA2取代，一个很重要的原因就是密码太短。

MAC层安全风险

一个未加密的802.11帧的加密步骤

- 计算未加密帧（包括帧头和负载）的CRC，即循环冗余校验码，把这个校验码称为ICV；
- 将负载和ICV连在一起，作为RC4算法的要加密数据RC4_Data；
- 选定一把WEP密钥，密钥编号为Key number，密钥为Key（WEP最多支持4把密钥）；
- 通过某种算法生成一个初始化向量IV（生成IV的算法可以任意），将IV与密钥Key连在一起，作为RC4算法的密钥RC4_Key；
- 将RC4_Data与RC4_Key代入RC4算法中，生成密文RC4_Result；
- 将未加密帧的帧头、IV、Key number、RC4_Result连在一起，就是加密后的帧。



WEP的工作模式与数据处理

MAC层安全风险

暴力破解具体方法

- 截获一个加密的802.11的数据帧，记下该帧的IV与负载的前5字节，将这5字节记为First5。
- 猜测密码是Key，将IV与Key相连，记为RC4_Key，然后把Data={0xaa,0xaa,0x03,0x00,0x00}这5个字节和RC4_Key代入RC4算法中，得到结果Result=RC4(Data,RC4_Key)，Result也是5个字节。如果Result与First5相等，那么Key就极有可能是正确的密钥。
- 穷举所有可能的Key，代入步骤2。

传输层安全风险

- **概述**

  简要总结传输层中的恶意活动，重点是TCP（传输控制协议）和UDP（用户数据报协议）攻击。

- **TCP（传输控制协议）**

  TCP是面向连接的传输协议，旨在支持数据包的可靠传输，通常用于传递电子邮件和将文件从一个网络节点传输到另一个网络节点。

- **UDP（用户数据报协议）**

  UDP是一种无连接传输协议，减少了协议开销和延迟，但作为一种代价，它无法保证可靠的数据传输。

传输层安全风险

TCP泛洪攻击&序列号预测攻击

泛洪攻击

攻击者向受害节点发送大量ping请求，例如ICMP回应请求，然后通过发送ping回复进行响应，例如ICMP echo回复。这将淹没受害节点的输入和输出缓冲区，并且当ping请求的数量足够高时，它甚至可能延迟其与目标网络的连接。

序列号预测攻击

- TCP序列预测技术是另一种TCP攻击，其试图预测发送节点的TCP分组的序列索引，然后制造该节点的TCP分组。
- TCP序列预测攻击者首先猜测受害发射机的TCP序列索引，然后使用预测的TCP索引制作分组，最后将其制作的分组发送到受害接收机。

传输层安全风险

UDP泛洪攻击

- 通过发送大量UDP数据包而不是TCP泛洪攻击中使用的ping请求来实现的 。
- UDP泛洪攻击者将大量UDP数据包发送到受害节点，受害节点将被强制发送大量回复数据包。受害节点将被恶意UDP数据包淹没，并被其他合法节点无法访问。
- 通过使用欺骗性IP地址将自己从合法节点隐藏起来，以生成恶意UDP数据包。通过限制UDP分组的响应速率来减轻这种UDP泛洪攻击的负面影响。可以使用防火墙来防御UDP泛洪攻击，以过滤掉恶意UDP数据包。

传输层安全风险

- **降级攻击**

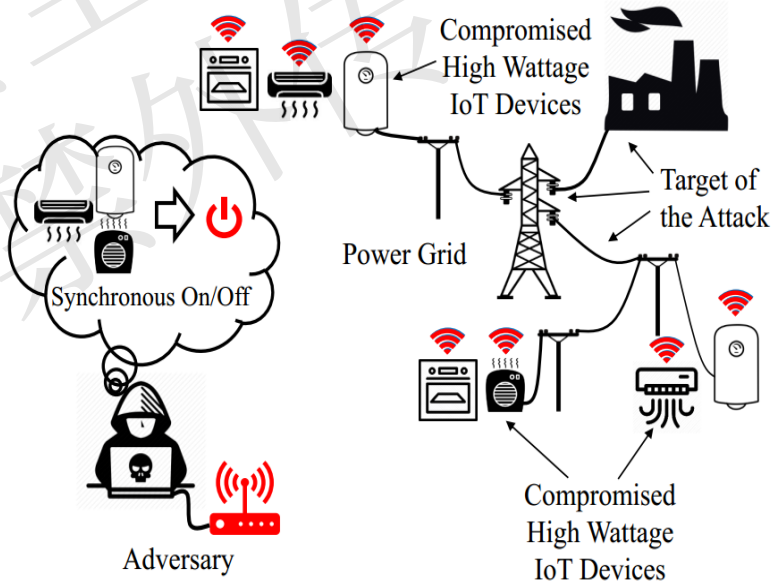
- **背景**：传输层身份验证包括SSL及其后续协议，即TLS协议。而针对传输层身份验证攻击最常见的是降级攻击，降级攻击是一种强制系统降级其安全性的攻击。

- **定义：** 攻击者使系统放弃安全性较高的工作方式，如加密连接，反而采用向下兼容的安全性较差的工作方式，如明文通信，它通常与加密攻击相关联。

传输层安全风险

攻击后果

- 相关风险漏洞可能会影响53亿智能设备，Android、iOS、Windows、Linux 系统的设备以及IoT设备等只要涉及到相关通信技术，不管是智能手机、笔记本电脑，还是智能电视、智能汽车或者其他IoT设备都有被攻击的风险。
- 通过攻击大面积物联网设备的齐开通关断可导致区域性停电，可见对物联网设备的攻击不单单对个体造成危害。



利用物联网设备攻击电网的示意图 (MadIoT)

4.3

通讯协议安全机制

通讯协议通讯安全机制

- 通讯协议设计安全机制
  - IEEE 802.15.4/ IEEE 802.15.4e
  - 6LoWPAN/RPL路由协议
  - CoAP应用层协议
- 通讯安全检测监测防护机制
  - 协议漏洞检测
  - 基于流量的安全监测
  - 基于辅助信道的安全协同认证

通讯协议设计安全机制

原理

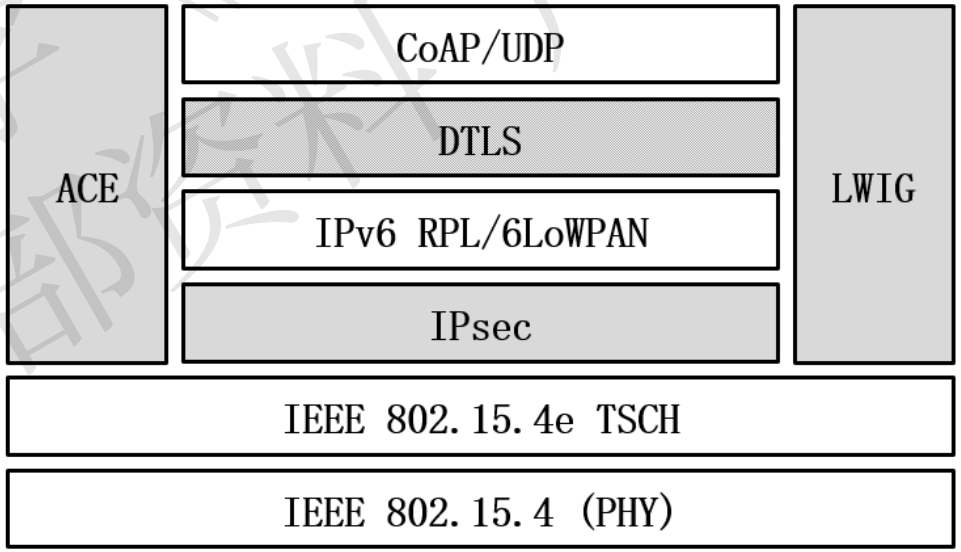
- **攻击预防（Attack Prevention）**：在真实攻击作用或影响目标系统之前所采用的预防安全机制。其中最重要的一项机制就是**访问控制**，即按访问对象身份及其所归属的定义组来限制对象对某些信息项的访问，或限制对某些控制功能的使用。

- **攻击规避（Attack Avoidance）**：不同于攻击预防机制，攻击规避主要面向攻击者可访问传输消息的情况，其目的是让攻击者无法有效篡改传输消息的内容。常用的攻击规避机制就是对传输消息进行预处理，即进行之前介绍的**加密机制**，如私钥机制、公钥机制和哈希算法等。

通讯协议设计安全机制

概述

- 物联网通信协议栈安全架构主要由通讯协议和其采用的安全协议组成。IEEE 802.15.4(e) 通过**安全域、安全等级和安全加密与认证**三方面提供安全服务，CoAP应用层协议采用**DTLS** (Datagram Transport-Layer Security) 提供端到端的安全传输，RPL网络层协议采用**IPSec**来保障安全。此外，ACE、DICE、LWIG等工作组正在开发用于资源受限设备的身份验证授权或加密机制。



通讯协议设计安全机制

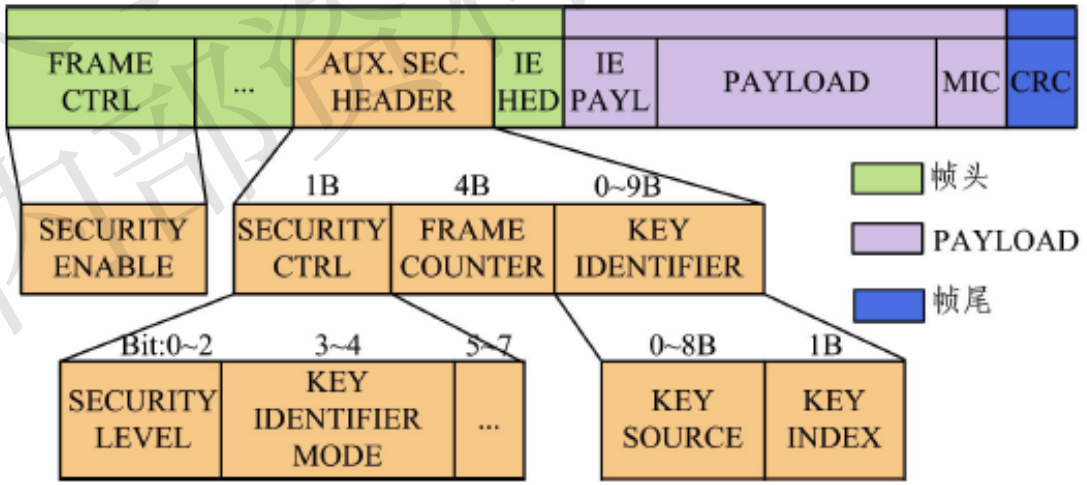
IEEE 802.15.4(e)

定义

- IEEE 802.15.4是物联网中的一种典型技术标准，其定义了**低速率无线个域网**（LR-WPAN）的协议，并规定LR-WPAN的物理层和媒体访问控制，**是Zigbee、6LoWPAN的基础**。IEEE 802.15.4标准在设计时充分考虑了安全问题，其在链路层提供了加密、认证以及重放保护等安全服务。 2012年IEEE 802.15.4工作组发布基于时间同步信道跳频（TSCH）的IEEE 802.15.4e标准，以面向短距离、低功耗的物联网应用场景。IEEE 802.15.4e的安全部分则直接继承了IEEE 802.15.4-2011标准。

通讯协议设计安全机制

- **IEEE 802.15.4(e)**
- 安全机制
  - **1) 附加安全域设置。**IEEE 802.15.4e帧主要由帧头、PAYLOAD帧和帧尾3部分组成。帧头由控制域、地址信息、附加安全头和IE头组成；PAYLOAD帧由IEPAYLOAD、数据部分和MIC组成，其都要参与加密运算；帧尾由2个字节的CRC校验码组成。附加安全域主要位于帧头部分，如下图所示。帧控制域包括**安全使能位**，当安全使能位为0时，表示无安全；当安全使能位为1时，表示设置安全，则需要在附加安全头字段设置相应**安全等级、密钥标识模式以及帧计数器**。



通讯协议设计安全机制

IEEE 802.15.4(e)

安全机制

2) 安全服务等级。IEEE 802.15.4e标准定义 8 个安全等级，包括无安全、仅认证MIC-32、仅认证MIC-64、仅认证MIC-128、仅加密ENC、加密与认证ENC-MIC-32、加密与认证ENC-MIC-64、加密与认证ENC-MIC-128。每种安全等级对应不同安全属性、数据保密性与完整性。

安全等级	安全属性	数据完整性	数据保密性
0	无	无	无
1	MIC-32	有	无
2	MIC-64	有	无
3	MIC-128	有	无
4	ENC	无	有
5	ENC-MIC-32	有	有
6	ENC-MIC-64	有	有
7	ENC-MIC-128	有	有

(MIC = Message.Integrity　　ENC = AES.Encryption)

通讯协议设计安全机制

- **IEEE 802.15.4(e)**
- 安全机制
  - **3) 安全加密与认证**：IEEE 802.15.4中的安全加密和认证是基于**AES-128加密机**。例如，当节点安全模式设置为ENC-MIC-32时，发送方先进行信息完整性认证,然后再进行加密操作；接收方收到数据包后先解密,然后再进行完整性认证。接收方的信息完整性认证的流程与发送方信息完整性的认证基本类似。

通讯协议设计安全机制

6LoWPAN协议

定义

- 6LoWPAN是一种基于IPv6的低速无线个域网标准，即IPv6 over IEEE 802.15.4。具体而言，6LoWPAN在IEEE802.15.4链路层之上提供IPv6网络访问服务，并通过简化IPv6包头来适应低功耗资源受限网络的要求。

安全机制

- 与其他协议不同，6LoWPAN标准中没有定义相关安全机制，但是相关RFC文档讨论了其安全问题，并指导协议设计中在IEEE 802.15.4链路层上使用AES/CCM 加密算法，在路由层上使用多路径路由算法或结合IPsec拓展机制来保障端到端通信的安全性。

通讯协议设计安全机制

IPSec

定义

- Internet Protocol Security（IPSec）是1） IETF制定的为保证在Internet上传送数据的安全保密性能的三层隧道加密协议； 2）通过对IP协议的分组进行加密和认证来保护IP协议的网络传输**协议族**（一些相互关联的协议的集合）。

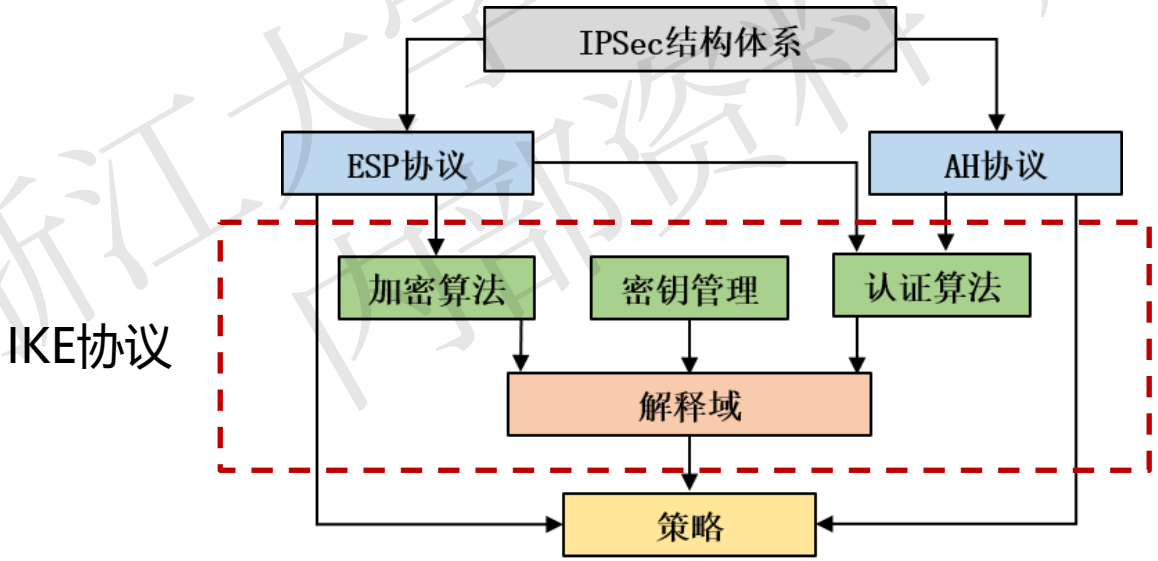
安全特点

- **数据机密性(Confidentiality)**：IPSec发送方在通过网络传输包前对包进行加密。
- **数据完整性(Data Integrity)**：IPSec接收方对发送方发送来的包进行认证，以确保数据在传输过程中没有被篡改。
- **数据来源认证(Data Authentication)**：IPSec接收方对IPSec包的源地址进行认证。这项服务基于数据完整性服务。
- **反重放(Anti-Replay)**：IPSec接收方可检测并拒绝接收过时或重复的报文。

通讯协议设计安全机制

- **IPSec**
- **安全机制体系结构**

IPsec主要由以下组成：1）**认证头(AH)**，是报文验证头协议，主要提供的功能有数据源验证、数据完整性校验和防报文重放功能；2）**封装安全载荷(ESP)**，是报文安全封装协议，将需要保护的用户数据进行加密后再封装到IP包中，保证数据的完整性、真实性和私有性；3）**安全关联(SA)**，提供算法和数据包，提供AH、ESP操作所需的参数。SA由**IKE(Internet Key Exchange)**建立和维护。

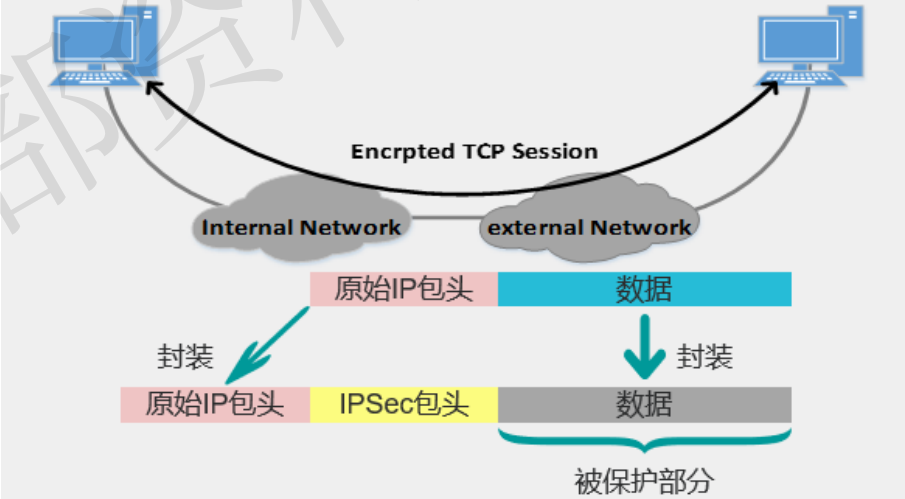


解释域：

规定了每个算法的参数要求和计算规则,及初始向量的计算规则等。

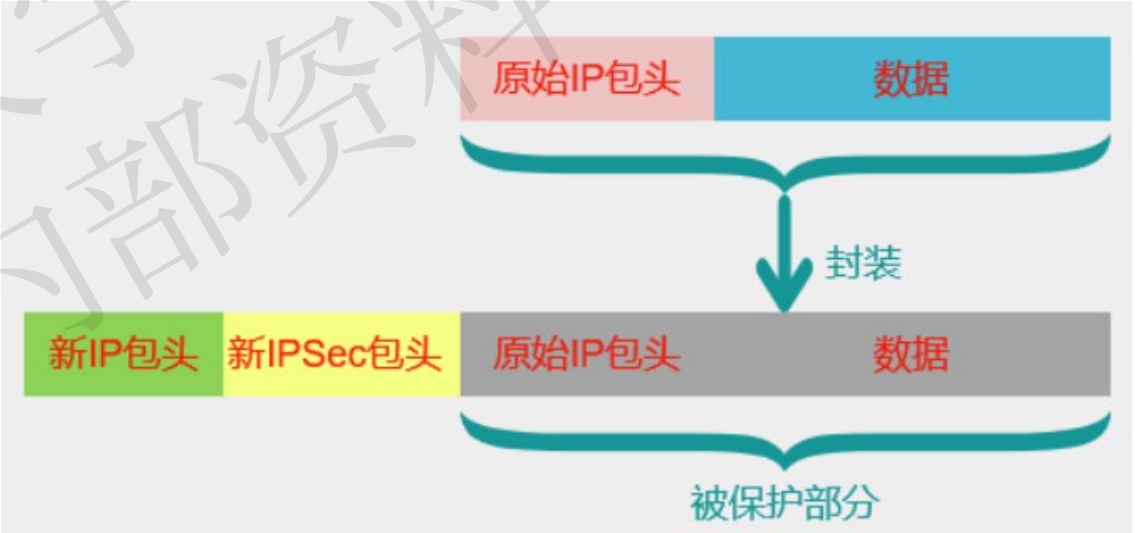
通讯协议设计安全机制

- **IPSec**
- **工作模式1：传输模式(Transport mode)**
  - **概述**：传输模式下，IPSec协议处理模块会在IP报头和高层协议报头之间插入一个**IPSec报头**。IP报头与原始IP分组中的IP报头是一致的，只是IP报文中的协议字段会被改成IPSec协议的协议号（50或者51），并重新计算IP报头校验和。传输模式保护数据包的有效载荷、高层协议，IPSec源端点不会修改IP报头中目的IP地址，原来的IP地址也会保持明文。传输模式只**为高层协议提供安全服务**。
  - **应用场景**：常用于主机和主机之间端到端通信的数据保护。



通讯协议设计安全机制

- **IPSec**
- **工作模式2：隧道模式(Tunnel mode)**
  - **概述**：在隧道模式下，原始IP分组被封装成一个**新的IP报文**，在内部报头以及外部报头之间插入一个IPSec报头，原IP地址被当作有效载荷的一部分受到IPSec的保护。通过对数据加密，还可以隐藏原数据包中的IP地址，这样更有利于保护端到端通信中数据的安全性。
  - **应用场景**：常用于私网与私网之间通过公网进行通信，建立安全VPN通道。



通讯协议设计安全机制

RPL路由协议

定义

- RPL就是一种适用于低功耗及有损网络(LLN网络)的距离矢量路由协议。

安全机制

- RPL协议设计中的安全机制体现在控制消息格式中，主要包括ICMPv6头部、安全域、DIO消息格式或DAO消息格式部分和可选域。RPL安全域可以设置**安全算法、密钥模式和安全等级**。安全算法主要包括加密、完整性认证（AES/CCM）和数字签名算法（SHA-256）。

ICMPv6头部		
Type（RPL包）	Code（DIO）	Checksum（校验）
RPL安全域		
DIO/DAO消息格式		
可选域		

通讯协议设计安全机制

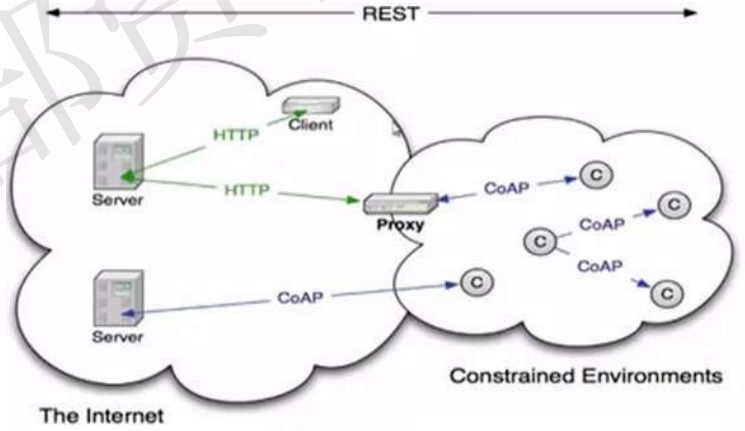
CoAP应用层协议

定义

- CoAP协议是基于REST架构与UDP传输协议的面向资源受限设备的物联网应用层协议。

安全机制

- CoAP协议大多数采用**DTLS（Datagram Transport-Layer Security）**来提供端到端的安全传输。DTLS(Datagram Transport Layer Security)即数据包传输层安全性协议是在现存**TLS**（Transport Layer Security）协议结构上，针对于UDP传输数据而设计的拓展安全机制。



通讯协议设计安全机制

SSL

定义

- SSL, 即Secure Sockets Layer安全套接字层，为Netscape所研发。SSL协议位于TCP/IP协议与各种应用层协议之间，为数据通讯提供安全支持。SSL协议分为两层：
- **SSL记录协议**（SSL Record Protocol）：建立在可靠的传输协议（如TCP）之上，为高层协议提供数据封装、压缩、加密等基本功能的支持，定义了传输的格式。
- **SSL握手协议**（SSL Handshake Protocol）：建立在SSL记录协议上，用于在实际的数据传输开始前，通讯双方进行身份认证、协商加密算法、交换加密密钥等。

通讯协议设计安全机制

SSL

加密套件

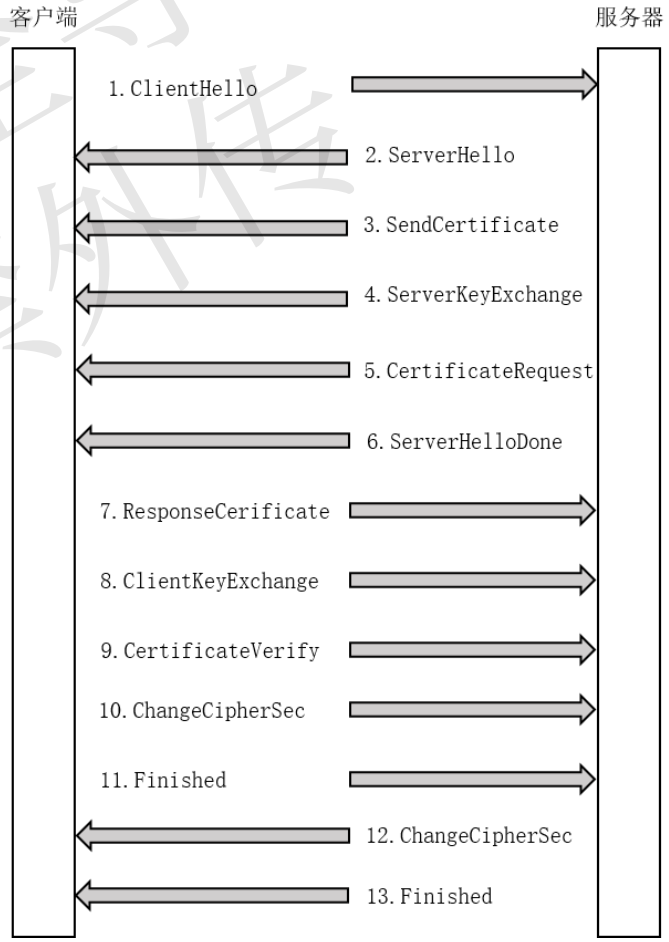
- **定义**：加密套件（CipherList）是指在ssl通信中，服务器和客户端所使用的加密算法的组合，由四个部分组成：
- **1）密钥交换**： 用于决定客户端与服务器之间在握手的过程中如何认证。SSL通常使用非对称加密算法来生成会话密钥，常用算法包括RSA，Diffie-Hellman，ECDH，PSK等；
- **2）加密算法**： 主要是对传输的数据进行加密传输。常用算法包括DES 56/56, RC2 56/128, RC4 128/128, AES 128/128, AES 256/256；
- **3）会话校验（MAC）算法**：为了防止握手消息本身被窜改。常用算法包括MD5，SHA等；
- **4）伪随机数函数。**

通讯协议设计安全机制

握手协议

概述

- 握手协议是客户机和服务器用SSL连接通信时使用的第一个子协议，握手协议包括客户机与服务器之间的一系列消息。该协议允许服务器和客户机**相互验证，协商加密和MAC算法以及保密密钥**，用来保护在SSL记录中发送的数据。握手协议是在应用程序的数据传输之前使用的。



通讯协议设计安全机制

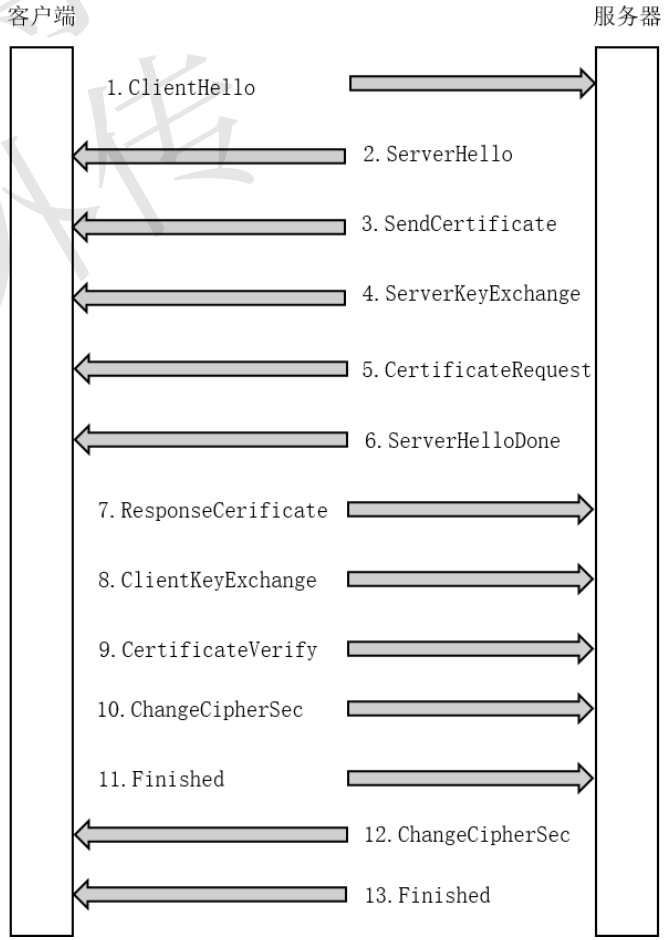
握手协议

Step1: ClientHello

- 1.支持的协议版本，比如TLS 1.0版
- 2.一个客户端生成的随机数，稍后用于生成对话密钥
- 3.支持的加密方法，比如RSA公钥加密（密码套件）
- 4.支持的压缩方法

Step2: ServerHello

- 1.确认使用的加密通信协议版本，比如TLS 1.0版本。如果浏览器与服务器支持的版本不一致，服务器关闭加密通信
- 2.一个服务器生成的随机数，稍后用于生成对话密钥
- 3.确认使用的加密方法，比如RSA公钥加密
- 4.服务器证书



通讯协议设计安全机制

握手协议

Step3: SendCertificate

- 服务器将数字证书发给客户端，使客户端能用服务器证书中的服务器公钥认证服务器。

Step4: ServerKeyExchange

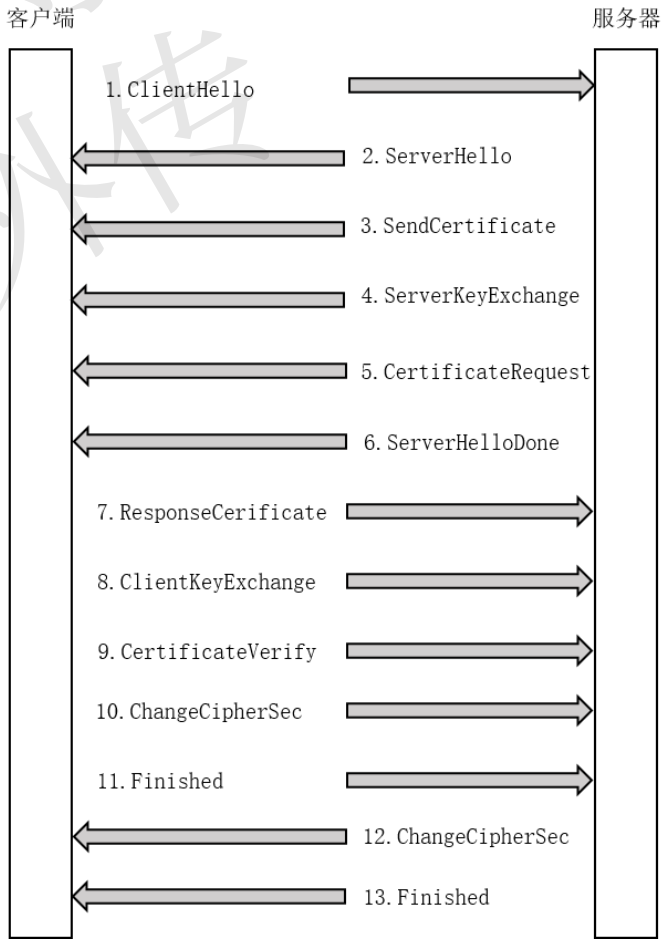
- **（此步可选）**在服务端向客户端发送的证书中没有提供足够的信息（证书公钥）的时候，还可以向客户端发送一个Server Key Exchange。

Step5: CertificateRequest

- 服务端可能会要求客户发送自身证书进行验证。

Step6: ServerHelloDone

- 服务器握手完成。



通讯协议设计安全机制

握手协议

Step7: ResponseCertificate

- （**此步可选**）客户发送自身证书信息给服务器证明身份。

Step8: ClientKeyExchange

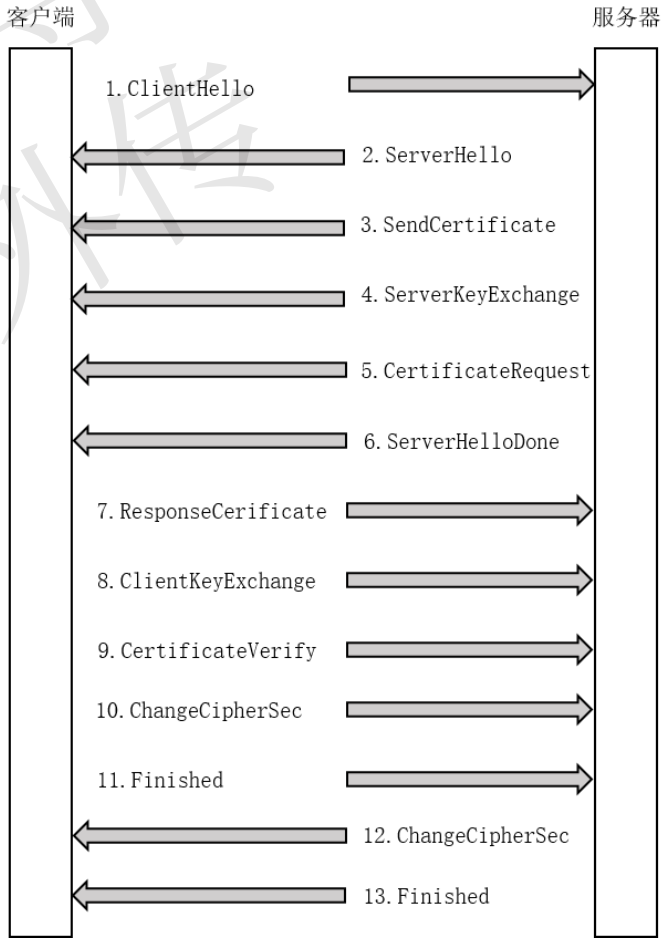
- 客户端利用服务器公钥加密预备主密钥，并发给服务器。

Step9: CertificateVerify

- （**此步可选**）客户端对预备秘密和随机数进行签名，证明拥有服务器证书的公钥。

Step10/12: ChangeCipherSec

- 表明发送端已取得用以生成连接参数的足够信息，已生成加密密钥，并且将切换到加密模式。



通讯协议设计安全机制

TLS

定义

- Transport Layer Security，即传输层安全性协议，前身是SSL，其用于在两个通信应用程序之间提供保密性和数据完整性。该协议由两层组成： TLS 记录协议（TLS Record）和 TLS 握手协议（TLS Handshake）。
- TLS的框架与SSL基本相同，但其在OSI模型的应用层和TCP/IP模型的传输层上运行，与高层的应用层协议**无耦合**。

通讯协议设计安全机制

SSL与TLS区别

- **1）版本号**：TLS记录格式与SSL记录格式相同，但版本号的值不同，TLS的版本1.0使用的版本号为SSLv3.1。
- **2）报文鉴别码**：SSLv3.0和TLS的MAC算法及MAC计算的范围不同。TLS使用RFC-2104定义的HMAC算法。SSLv3.0使用了相似的算法，两者差别在于SSLv3.0中，填充字节与密钥之间采用的是连接运算，而HMAC算法采用的异或运算。但是两者的安全程度是相同的。
- **3）伪随机函数**：TLS使用了称为PRF的伪随机函数来将密钥扩展成数据块，是更安全的方式。
- **4）报警代码**：TLS支持几乎所有的SSLv3.0报警代码，而且TLS还补充定义了很多报警代码，如解密失败（decryption_failed）、记录溢出（record_overflow）、未知CA（unknown_ca）、拒绝访问（access_denied）等。
- **5）密文族和客户证书**：SSLv3.0和TLS存在少量差别，即TLS不支持Fortezza密钥交换、加密算法和客户证书。

SSL与TLS区别

- **6） CertificateVerify和Finished消息**：SSLv3.0和TLS在用CertificateVerify和Finished消息计算MD5和SHA-1散列码时，计算的输入有少许差别，但安全性相当。
- **7）加密计算**：TLS和SSLv3.0在计算主密钥时采用的方式不同。
- **8）填充**：用户数据加密之前需要增加的填充字节。在SSL中，填充后的数据长度哟啊达到密文快长度的最小整数倍。而在TLS中，填充后的数据长度可以是密文块长度的任意整数倍（但填充的最大长度为255字节），这种方式可以防止基于对报文长度进行分析的攻击。

通讯协议设计安全机制

DTLS

定义

- Datagram Transport Layer Security，即**数据包传输层安全性协议**。TLS不能用来保证UDP上传输的数据的安全，因此DTLS试图在现存的TLS协议架构上提出扩展，使之支持UDP，即成为TLS的一个支持数据包传输的版本。

与TLS区别

- DTLS与TLS最大的区别在于**握手协议**。由于DTLS面向UDP场景，所以其必须依靠**重传机制**来保证握手信息的可靠传递，且利用的重传机制必须简单和轻量，以提升可移植性。因此，DTLS协议在UDP提供的socket之上实现了客户机与服务器双方的握手连接，在握手过程中通过使用**PSK或ECC**实现了加密，并利用**cookie验证机制**和**证书**实现了通信双方的身份认证，以及用在报文段头部加上序号，缓存乱序到达的报文段和重传机制实现了可靠传送。

通讯安全检测监测防护机制

协议漏洞检测

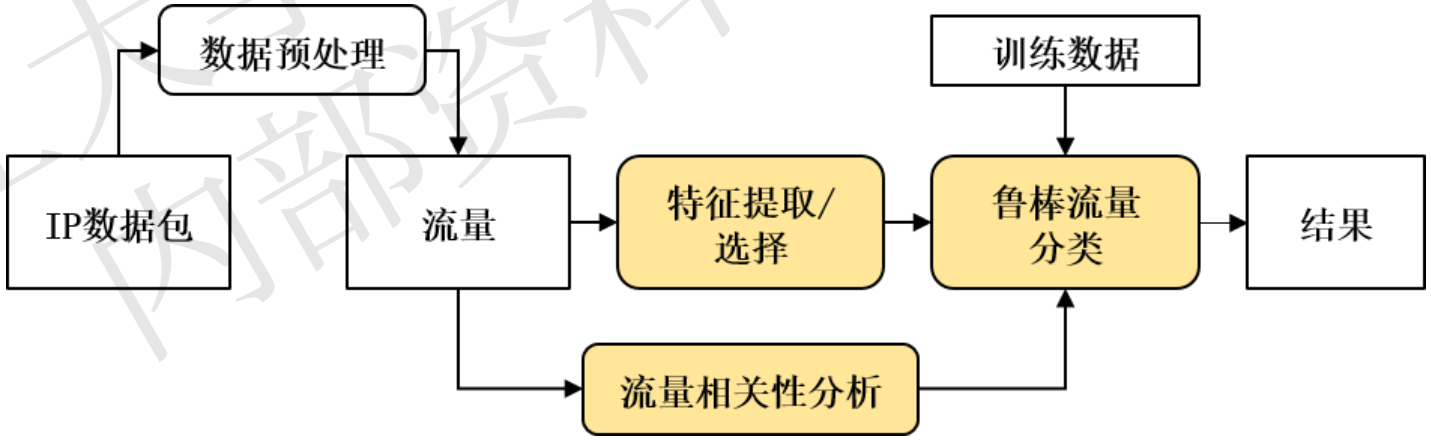
- **基于静态代码分析的协议漏洞检测：**根据通讯协议编写**源代码**，查找其设计缺陷和潜在漏洞。
- **基于动态污点追踪的协议漏洞检测：**分析通讯协议执行过程中信息流或控制流经历的所有可能路径，然后**设置污点**并追踪**敏感信息流的完整性与泄露路径**，从而实现通讯协议的漏洞挖掘。
- **基于攻击推演的协议漏洞检测：**通过对目标待测协议执行常见**网络安全攻击**以检测协议设计中是否存在漏洞。
- **基于符号执行的协议漏洞检测：**将协议执行的具体输入**符号化**，并将这些符号变量作为输入进行执行。然后在给定的**约束**内，探索尽可能多的**执行路径**，检查协议中是否存在漏洞。
- **基于模糊测试的协议漏洞检测：**通过向被测目标输入大量的**非预期数据**并监测其异常结果来发现漏洞。

通讯安全检测监测防护机制

基于流量的安全监测

基于流量的安全监测技术指的是通过将通讯链路中传输流量进行实时监控以实现网络安全管理的技术，其核心是**流量分类**。基于流量的安全监测技术可以实现服务质量（QoS）控制、合法拦截和入侵检测。执行流程一般为：

(1) 数据捕获 → (2) 构造流量 → (3) 提取特征 → (4) 流量表征 → (5)流量分类 → (6)安全分析

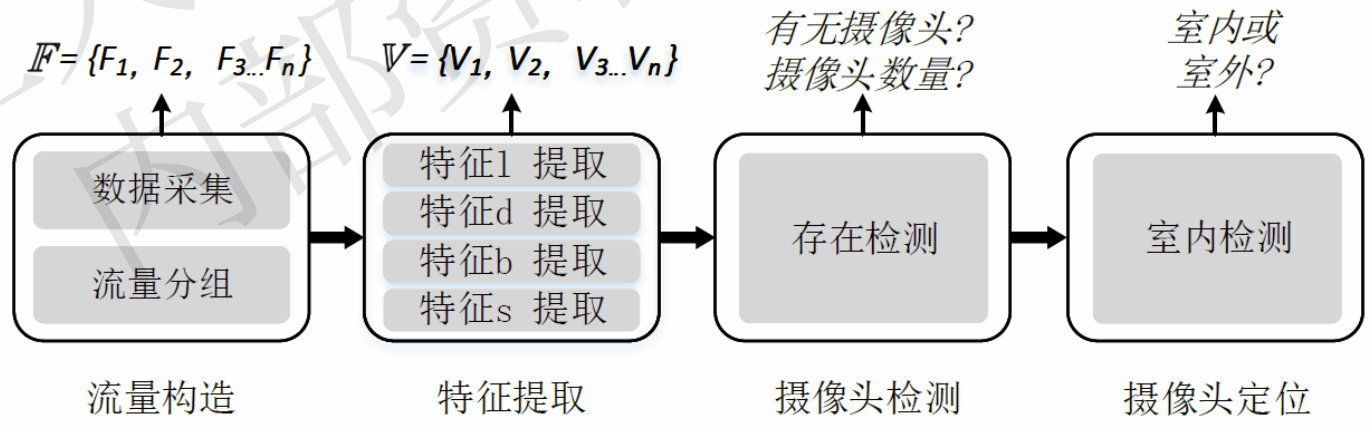


基于流量相关性分析的安全监测算法流程图

通讯安全检测监测防护机制

基于智能手机的隐藏无线摄像头检测 — DeWiCam

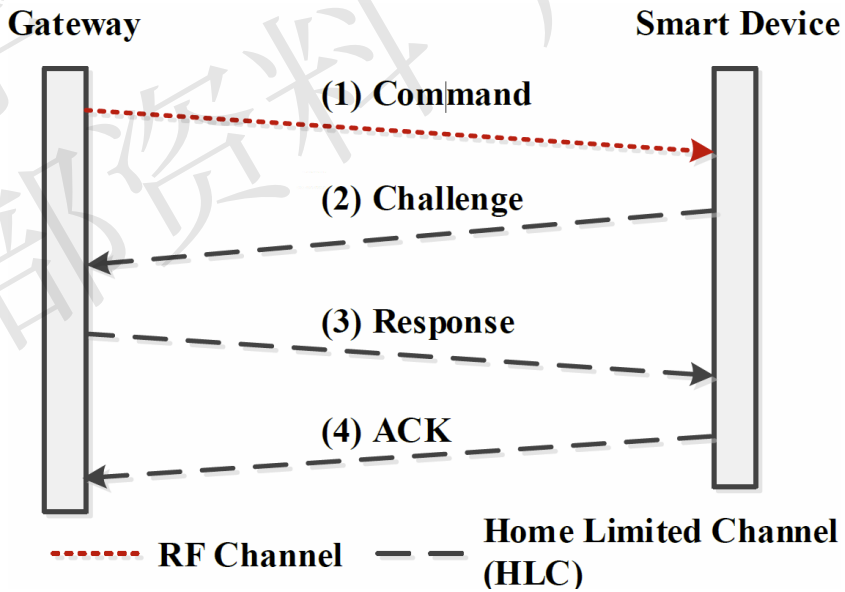
- **概述**：DeWiCam 使用智能手机采集、分析当前空间的无线网络流量，从而检测无线网络摄像头的存在，并通过人为干预，确定该无线网络摄像头是否处于当前房间。
- **基本原理**：无线摄像头在工作时不断产生由视频和音频组成的无线网络流量。由于其特殊的工作原理，无线摄像头与其他网络应用具有不同网络流量特征，如数据包长度分布等。通过分析识别网络流量上存在的特征，DeWiCam能够识别无线摄像头网络流量，从而检测无线摄像头的存在。



通讯安全检测监测防护机制

基于辅助信道的安全协同认证

基于辅助信道的安全协同认证是指通过引入辅助信道（辅助信道上的传输信号具有**边界衰减性**、**不可察觉性**等特点，如红外光、超声波等），结合**挑战响应**机制，实现对通信双方身份的合法性验证。例如，当网关向设备发送一个无线电控制指令时，设备会在辅助信道上向网关发起身份挑战，后者依据挑战做出相关响应。若响应正确，则设备执行无线电控制指令内容。



小节

- ❖软件概念、固件结构
- ❖软件漏洞概述
- ❖软件安全防护

- ❖通讯协议定义与分类
- ❖通讯协议安全风险
- ❖通讯协议安全机制