物联网设备固件漏洞利用与攻击缓解绕过

实验背景

近年来物联网行业发展迅速，物联网设备广泛应用于人类生活的各个领域如智能家居、智慧医疗等。甚至越来越多的之前长期处于闭源环境下的工业、车辆以及军用设备和关键基础设施等嵌入式设备逐步接入互联网成为物联网设备。这些无处不在的物联网设备构成了一个万物互联的巨大网络，极大地促进了人类社会的信息化、智能化发展。根据全球移动通信系统协会（GSMA）数据，2019-2022 年全球嵌入式设备复合增长率高达 12.7%，2022 年全球嵌入式设备已达到 172 亿台，全球物联网市场规模达到 61344 亿元。

物联网设备核心大多基于微控制器如 MIPS、ARM Cortex-M 等，不同于高性能处理器架构如 x86，其功耗小、成本低，架构简单。但与此同时，其计算性能低、存储空间有限，同时缺乏必要的硬件安全模块如 MMU，导致其难以设备上直接部署传统攻击缓解技术如 ASLR 等。另一方面，运行于物联网设备的软件与系统需要对各式各样的外围硬件设备进行交互来实现特定功能，并且实时性要求高。因此，基于微控制器的物联网设备的软件与系统大多由低级语言开发如 C 和 C++等，并且通常将驱动、应用软件和系统静态编译为一个统一的整体（通常称为“固件”），导致其难以避免产生内存错误。介于物联网设备软硬件特点，以及现阶段开发人员和厂商缺乏必要的安全意识与安全测试，导致物联网设备固件中存在大量易被攻击者利用的漏洞。这些漏洞不仅会对物联网产品造成破坏，甚至对用户、社会乃至国家造成严重危害。

实验目的：

面对层出不穷的物联网设备漏洞与攻击事件，作为网络空间安全专业的学生了解和掌握常见的物联网设备漏洞发现与利用方法以及利用常见的硬件特性设计简单的系统防御措施显得十分重要。

实验准备：

实验工具：

物联网设备固件逆向工具：IDA Pro

物联网设备固件模拟器：QEMU (5.0.0 以上版本均可)

注意：仅安装 qemu-system-arm 即可

源码安装：https://www.qemu.org/

配置命令：

./configure --prefix=/root/Downloads/qemu-7.0.0-rc0/build --target-list=arm-softmmu --enable-debug

Make

Make install

也可以直接采用 apt 安装

参考学习资料书籍：

ARM Cortex-M3 M4 权威指南（资料已上传超星平台）

IDA Pro 权威指南 2

FreeRTOS 源码讲解与应用开发

Mastering the FreeRTOS™ Real Time Kernel (https://freertos.org/fr-content-

src/uploads/2018/07/161204_Mastering_the_FreeRTOS_Real_Time_Kernel-A_Hands-On_Tutorial_Guide.pdf)

背景知识

1. ARM 架构基础 **权威指南第四章**
   a) 操作模式和状态
   b) 寄存器
   c) 存储系统
   d) 异常和中断
   e) 系统复位流程
2. ARM 指令集 **权威指南第五章**
   a) 存储器访问指令
3. ARM 存储器内存保护单元 **权威指南第九章**

实验内容与分数占比：

1. 题号 1：物联网设备溢出漏洞利用 40%
2. 题号 2：利用溢出漏洞实现在 FreeRTOS MPU V10.4 版本的系统提权和 Flag 函数打印 40%
3. 题号 3：利用溢出漏洞实现在 FreeRTOS MPU V10.4 版本的系统提权和 Flag 函数打印 20%

注：Flag 提交和实验报告各占比百分之五十。但实验报告和 flag 提交要对应，如果 flag 提交了但实验报告没有对应题目的解析或者完全不对那么这道题无法得分。

4. 附加思考题答对每题加 2 分

免修同学想获得更高分数必须作答实验三

Flag 提交平台：

信安系：http://222.20.126.138:8088/

网安系：http://222.20.126.138:8080/

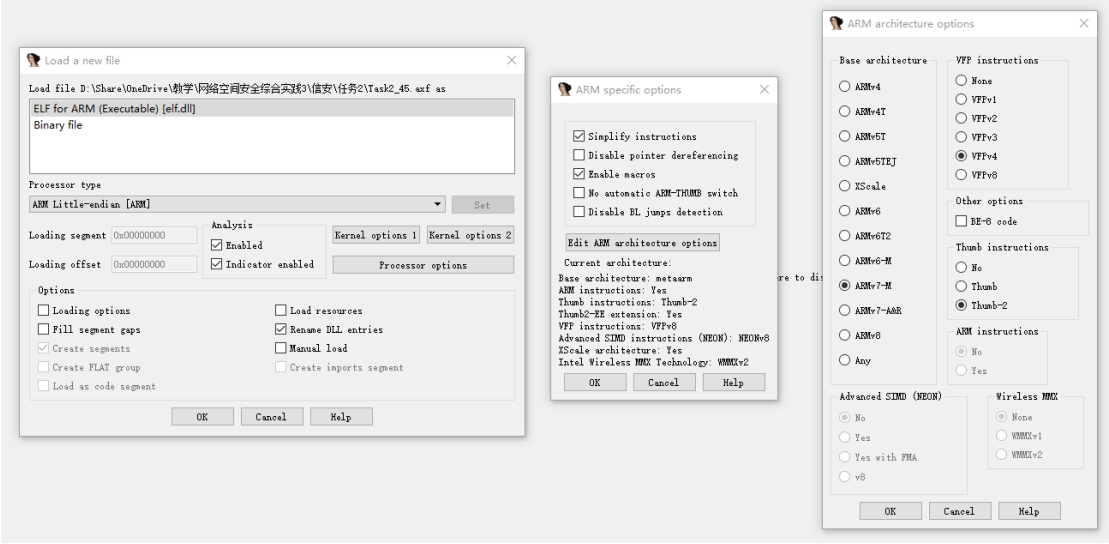
实验 1 物联网设备溢出漏洞利用

实验要求：

找到固件中可以溢出的缓冲区 buffer 和目标 flag 函数，利用溢出攻击覆盖返回地址打印 Flag 函数。

实验步骤：

1. 选择 ARM v7m 架构利用 IDA Pro 根据尾号正确加载实验 1 所需固件 task1_XX.axf



2. 通过逆向固件找到可以溢出的缓冲区
3. 找到打印 flag 的函数地址(包含了 flag 字符)
4. 构造 shellcode 使其溢出返回地址到 Flag 函数
5. 模拟执行指令：./qemu-system-arm -M mps2-an386 -cpu cortex-m4 -m 16M -nographic -d in_asm,nochain -kernel task2_xx.axf -D log.txt

依次填写学号后四位、总 buffer 长度、从低地址到高地址 buffer 每个 Byte 的十六进制数，每次输入完后请输入回车确认。

注意：未获取 flag 请关闭终端或杀死 qemu 进程再重新尝试。

实验 1 在报告中提交的内容包括：

1. IDA 加载过程测试固件的过程
2. 存在溢出缓冲区函数和 Flag 函数截图以及发现步骤
3. 栈原始布局和溢出示意图以及原理比如长度和返回地址要标注
4. 模拟执行获取 flag 的截图

实验 2 利用溢出漏洞绕过 FreeRTOS-MPU (v10.4.4) 保护

实验要求：

FreeRTOS 提供了基于 MPU 的内存保护方案，本实验中其设置 MPU 布局如下表所示：

区域编号	地址范围	访问属性
0	代码段	只读，可执行
1	0x00-0x00008000 RTOS kernel API	特权只读，可执行， 非特权不可访问
2	SysCall 系统调用段	全部只读可执行
3	0x2000000-0x20005000 RTOS kernel data	特权读写，可执行 非特权不可访问
4	任务栈地址范围	非特权级读写，不 可执行
5-7	自定义区域1	自定义

利用溢出漏洞实现在 FreeRTOS MPU V10.4.4 版本的系统提权和 Flag 函数打印

实验步骤

1. 利用 IDA pro 加载实验的 task3b_xx.axf 文件（参考实验 1）
2. 找到可以用于提权和 Flag 打印的函数或指令地址
3. 找到利用的溢出缓冲区
4. 模拟执行 task2_xx.axf 文件

   模拟执行指令：./qemu-system-arm -M mps2-an386 -cpu cortex-m4 -m 16M -nographic -d in_asm,nochain -kernel task2_xx.axf -D log.txt

   依次填写学号**后四位**、**总 buffer 长度**、从低地址到高地址 buffer 每个 Byte 的十六进制数，每次输入完后请输入回车确认。

   注意：未获取 flag 请关闭终端或杀死 qemu 进程再重新尝试。

实验 2 在报告中需要提交的内容包括：

1. 打印 Flag 函数名称和地址
2. 用于提权的函数名称和地址
3. 存在在溢出的缓冲区、栈示意图、溢出提权和覆盖返回地址的解析过程
4. 模拟运行截图

实验 3 利用溢出漏洞绕过 FreeRTOS-MPU(v10.4.5)保护

FreeRTOS 提供了基于 MPU 的内存保护方案，本实验中其设置 MPU 布局如下表所示：

区域编号	地址范围	访问属性
0	代码段	只读，可执行
1	0x00-0x00008000 RTOS kernel API	特权只读，可执行， 非特权不可访问
2	SysCall 系统调用段	全部只读可执行
3	0x2000000-0x20005000 RTOS kernel data	特权读写，可执行 非特权不可访问
4	任务栈地址范围	非特权级读写，不 可执行
5-7	自定义区域1	自定义

利用溢出漏洞实现在 FreeRTOS MPU V10.4.5 版本的系统提权和 Flag 函数打印。注意：该版本的 MPU 系统调用实现修复了实验二中的问题，但仍然存在其他安全问题可以被利用。

实验步骤

1. 利用 IDA pro 加载实验的 task3b_xx.axf 文件（参考实验 1）
2. 找到可以用于提权和 Flag 打印的函数或指令地址
3. 找到利用的溢出缓冲区
4. 模拟执行 task2_xx.axf 文件

   模拟执行指令：./qemu-system-arm -M mps2-an386 -cpu cortex-m4 -m 16M -nographic -d in_asm,nochain -kernel task2_xx.axf -D log.txt

   依次填写学号**后四位**、**总 buffer 长度**、从低地址到高地址 buffer 每个 Byte 的十六进制数，每次输入完后请输入回车确认。

   注意：未获取 flag 请关闭终端或杀死 qemu 进程再重新尝试。

实验 3 在报告中需要提交的内容包括：

1. 打印 Flag 函数名称和地址
2. 用于提权的函数名称和地址
3. 存在在溢出的缓冲区、栈示意图、溢出提权和覆盖返回地址的解析过程
4. 模拟运行截图

总体需要提交的实验报告内容：

1. 实验环境
2. 每个实验需要提交的内容
3. 实验心得和反馈 (+2 分)

思考题：(不占总成绩，答对每个小题加 2 分)

a) MPU 和 MMU 对于内存保护上的主要差别是什么？简述 ARM、RISC-V 或者 MIPS 上除了 MPU 以外的其他可用于系统保护的硬件特性？
b) 如何利用 MPU 实现对任务栈的溢出保护？请描述设置方法和简要步骤。
c) 基于 ARMv7m 架构谈一谈安全的物联网操作系统的系统调用设计与实现。
d) 基于 ARMv7m 架构谈一谈一个函数中存在缓冲区溢出是否就一定可以被利用呢？
e) 对于上述实验能否用动态调试的方法从内存中获取 flag？如果可以请描述具体步骤以及运行获取 flag 截图。