1 前言

* 1. 目的

软件开发的过程中需要对项目的进度进行跟踪和监控，明确组员的任务与职责，并协调组内员工的工作，因此需要安排后期工作计划。

* 1. 术语与缩略语

|  |  |  |
| --- | --- | --- |
| 编号 | 术语 | 说明 |
| 1 | 报文 | 报文(message)是网络中交换与传输的数据单元，即站点一次性要发送的数据块 |
| 2 | pcap | pcap 是packet capture 的缩写，pcap文件格式是常用的数据报存储格式 |
| 3 | 接口 | API，是一些预先定义的函数，目的是提供应用程序与开发人员基于某软件或硬件得以访问一组例程的能力，而又无需访问源码，或理解内部工作机制的细节。 |
| 4 | 端口 | 是一种经由软件创建的服务，在一个计算机操作系统中扮演通信的端点 |
| 5 | 端口监听 | 端口监听是指对客户端（个人机器）所操作的一种信息记录。端口监听还用于实现对共享目录访问的监测和控制 |

1. 产品简介
   1. 产品概述

Scapy 是一个用于计算机网络的数据包处理工具，最初由Philippe Biondi用python编写。它允许用户能够发送，嗅探和剖析并伪造网络数据包。与此同时，它也允许用户构建可以探测，扫描或攻击网络的工具。简单来讲，Scapy是一个功能强大的交互式数据包操作程序。

Scapy能够伪造或者解码大量协议的数据包，在线路上发送和捕获数据包，并匹配请求和回复。它同样也能轻松处理扫描、路由跟踪、探测、单元测试、攻击和网络发现等任务。从功能上讲，Scapy可以取代hping，arpspoof，arp-sk，arping，p0f甚至是Nmap，tcpdump和tshark的某些功能模块。

Scapy的特点有如下几点：

可以完整地返回它收到的所有信息。

可以自由地构建所需的数据包，对数据包进行填充、任意修改字段值，并按照需要进行堆叠。

支持大量报文的解析和协议的构建。

* 1. 产品接口简要分析

• 插件调用目录：arch

Scapy的部分功能在实现时需要使用其他插件，比如tcpdump、winpcap等，同时也需要获取当前系统的一些硬件信息，arch目录下就封装了实现这些功能的各种模块。

• 网络标准目录：asn1

在计算机网络领域，编码、传输、解码采用的是ASN.1(Abstract Syntax Notation one)标准，asn1目录下封装了实现该标准的相应模块。

• 扩展模块目录：contrib

由非官方开发者开发的扩展功能模块，包括新的协议模块和一些扩展功能模块，经过官方审核测试之后，就好放在contrib目录下，在使用时可以显示的去调用这些模块。

• 报文格式目录：layers

Layers目录下封装了Scapy所支持的所有协议的具体格式，包括所有协议字段名称、数据类型、大小、位置等信息，以及一些必要的解码方式。

• 基本功能模块目录：modules

Scapy的很多功能在实现时都需要调用一些共同的比较底层的基本模块，modules目录下就封装了需要显示加载的各种基本功能模块。

• 测试工具目录：tools

tools目录下包含为了测试各部分功能而准备的功能模块。

• 配置模块 ：config.py

Scapy在启动时会调用该模块，获取运行环境信息，初始化其他功能模块所需的配置参数。

• 协议字段格式模块：field.py

该模块封装了各种类型字段的格式信息，包括字段数据类型，大小等。

• 报文对象模块：packet.py

该模块封装了报文对象所拥有的各种参数和方法，所有的协议模块都继承于此模块。

• 报文发送/接收/嗅探模块：sendrev.py

该模块封装了与报文发送、接收和嗅探相关的所有功能函数。

• 工具库模块：utils.py

该模块封装了Scapy中常用的一些功能，比如pcap文件的读取和写入、数据类型转换、插件调用等。插件调用目录：arch

Scapy的部分功能在实现时需要使用其他插件，比如tcpdump、winpcap等，同时也需要获取当前系统的一些硬件信息，arch目录下就封装了实现这些功能的各种模块。

网络标准目录：asn1

在计算机网络领域，编码、传输、解码采用的是ASN.1(Abstract Syntax Notation one)标准，asn1目录下封装了实现该标准的相应模块。

扩展模块目录：contrib

由非官方开发者开发的扩展功能模块，包括新的协议模块和一些扩展功能模块，经过官方审核测试之后，就好放在contrib目录下，在使用时可以显示的去调用这些模块。

报文格式目录：layers

Layers目录下封装了Scapy所支持的所有协议的具体格式，包括所有协议字段名称、数据类型、大小、位置等信息，以及一些必要的解码方式。

基本功能模块目录：modules

Scapy的很多功能在实现时都需要调用一些共同的比较底层的基本模块，modules目录下就封装了需要显示加载的各种基本功能模块。

测试工具目录：tools

tools目录下包含为了测试各部分功能而准备的功能模块。

配置模块 ：config.py

Scapy在启动时会调用该模块，获取运行环境信息，初始化其他功能模块所需的配置参数。

协议字段格式模块：field.py

该模块封装了各种类型字段的格式信息，包括字段数据类型，大小等。

报文对象模块：packet.py

该模块封装了报文对象所拥有的各种参数和方法，所有的协议模块都继承于此模块。

报文发送/接收/嗅探模块：sendrev.py

该模块封装了与报文发送、接收和嗅探相关的所有功能函数。

工具库模块：utils.py

该模块封装了Scapy中常用的一些功能，比如pcap文件的读取和写入、数据类型转换、插件调用等。

* 1. 产品用户定义

Scapy的用户定义为开发人员。

* 1. 产品优化方向

本项目将对Scapy的功能进行优化，分为以下三个部分。

首先是文件切分功能。在pcap文件中，报文是使用字节码存储的，所占用的存储空间是非常小的。但解析后的每条报文包含很多对象、字段和方法，因此解析后的数据所占用的存储空间可能是原始报文大小的数十倍。这对内存来说是一个极大的挑战，并且内存占用过多对程序的运行效率也有极大的负面影响。因此我们将对较大的pcap文件进行切分处理。

其次，本项目计划为Scapy添加嗅探存储功能。Scapy本身的嗅探模块sniff()提供了非常丰富的功能，包括BPF筛选规则、自定义筛选规则、针对每条报文的自定义处理函数等，但其在设计上有一个很大的缺陷。当使用者需要获取所嗅探到的报文数据时，sniff()模块会将所有嗅探到的符合要求的报文进行解析处理后保存在内存中，等到嗅探结束时再返回给使用者。这就导致sniff()模块无法处理大规模的报文流量，否则会发生内存泄漏进而被系统关闭进程。为了解决这个问题，本项目计划在嗅探的过程中分批存储截获到的报文，并及时释放内存。

另外，本项目计划添加协议类型获取功能。在解析报文时，Scapy会根据协议层次划分一层一层的进行解析。解析结果也是层层堆叠，每一层都被封装成一个单独的对象，该对象拥有一个payload属性用于保存下一层协议的对象实例。在处理多协议的流量数据时，使用者通常需要根据报文的协议类型对报文进行不同的处理操作。但在解析后的报文中，用户无法直接获取到协议类型。因此，Scapy添加协议类型获取功能后将为用户提供极大的方便。

* 1. 产品技术路线

为了实现对Scapy的优化与完善，我们拟定了如下的技术路线。首先利用Scapy的内置函数wrpcap()和PcapReader()实现pcap文件的切分。其次，为了不影响已有嗅探模块，我们准备单独封装一个模块sniff\_store()，该模块在原有的sniff()模块基础之上增加两个参数batch\_count和res\_path，用来指定每次存储时的报文个数以及pcap文件的存放目录。最后，设计一个逻辑，获取经过解析的报文类型。

1. 项目总体实施计划
   1. 参加人员

|  |  |  |
| --- | --- | --- |
| 职位 | 姓名 | 性别 |
| 组长 | 李铎坤 | 男 |
| 组员 | 陈鸿超 | 男 |
| 组员 | 刘颖 | 女 |
| 组员 | 袁梦阳 | 男 |

* 1. 完成项目最后期限

本学期校历第十七周

* 1. 本计划的批准者与批准日期

批准者：任健老师

批准日期：校历第二周

* 1. 项目分工原则

本项目的分工依据为组员的个人特点。在确定项目之后，组内开展了多次会议，并在会议上阐述了个人优势。组长综合了个人特点以及个人意愿进行了任务划分，经过集体讨论之后，最终的任务分配见第四章。

* 1. 组织形式

1. 每周2-3次线下会议，配合1-2次线上交流，讨论本周的任务划分、当前工作进展以及目前遇到的问题。
2. 建微信和QQ群，随时保持交流。
3. 通过华为云和github实现资源的共享。
   1. 各阶段交付的产品

|  |  |  |
| --- | --- | --- |
| **序号** | **产品名称** | **交付时间** |
| 1 | 需求规格说明书 | 第三周 |
| 2 | 软件项目计划说明书 | 第四周 |
| 3 | 项目计划（华为云项目规划） | 每周更新 |
| 4 | 每周会议记录 | 每周更新 |
| 5 | 需求检查单 | 第六周 |
| 6 | 软件评审报告 | 待定 |
| 7 | 软件产品改进与展示 | 第十一周 |
| 8 | 软件测试分析报告 | 第十二周 |
| 9 | 软件测试评审报告 | 待定 |
| 10 | 软件进度分析报告 | 待定 |
| 11 | 工作量分析报告 | 每周更新 |
| 12 | 配置管理 | 每周更新 |

* 1. 进度安排原则

进度安排的原则是首先要估算项目规模大小，然后估算实现这些规模要付出的代价，最后基于这种估算制定项目进度计划。

对于可能出现的进度变化，小组计划在每周六的早晨进行讨论。对下周的任务进行明确、细化，如果需要的话再进行微调，并在华为云上进行项目规划的更新。

* 1. 迭代划分情况

根据实验要求和项目实际情况，本项目的迭代划分计划为：

* 第一个迭代：1-4周，工作内容为项目准备、需求分析、配置管理、工作量分析等。
* 第二个迭代：5-6周，工作内容为需求评审和复评审。
* 此后每周为一个迭代，工作基于该周的主要工作，并更新配置管理和工作量分析等。
  1. 遵循标准

本项目遵从以下标准：

GB/T 13702-1992 计算机软件分类与代码

GB/T 19003-2008 软件工程

GB/T 9386-2008 计算机测试文档编制

GB/T 9385-2008 计算机软件需求规格说明

GB/T 5532-2008 计算机软件测试规范

GB/T 8567-2006 计算机软件文档编制规范

* 1. 硬件环境

CPU：暂无要求，主流性能即可

内存：4G内存及以上（系统最低2GB，Scapy报文解析较占用内存空间）

硬盘：20G硬盘及以上（windows 7 64位安装最低配置）

* 1. 软件环境
* 操作系统：

Windows 7及以上： 已安装Npcap/Winpcap，

Ubuntu 16.04及以上： 已安装tcpdump

* Anaconda（可创建多版本Python虚拟环境），Python 2.7+，Python 3.4+
* Pip库：matplotlib，pyx，vpython，cryptography等