

基于Peach的协议测试设计与实现

赵丽娟, 温巧燕, 张华

(北京邮电大学网络与交换技术国家重点实验室, 北京 100876)

摘要: 计算机技术、网络技术的快速发展, 加速了社会信息化的进程, 使人们与信息系统的关系越来越密切。在庞大的信息社会, 安全性问题已经成为信息系统的主要问题之一。及早开展安全测试, 有助于在很大范围内解决安全问题。本文首先介绍Fuzz测试原理, 然后介绍基于框架的Fuzz测试工具Peach, 其次分析HTTP协议, 最后设计并实现Peach对HTTP协议的测试。

关键词: Fuzz测试; HTTP协议; Peach

中图分类号: TP393.8

Peach-based Protocol Testing design and implementation

ZHAO Lijuan, WEN Qiaoyan, ZHANG Hua

(State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876)

Abstract: With the great development of computer and network technology, accelerate the process of society informatization, brought people into closer relations with the information system. Security has become one of the main problems in information society. Make security testing as early as possible and assures the security problem was sloved. This paper first introduce the principle of fuzz testing, then introduce fuzz testing tool Peach which is based on framework, next analyze HTTP protocol, last design and implement the testing for HTTP protocol by Peach.

Key words: Fuzz testing; HTTP protocol; Peach

0 引言

传统的Fuzz指的是一种非常简单的黑盒测试技术或随机测试技术, 用来发现软件的缺陷(flaws)。1990年Miller等人^[1]发现, 通过简单的Fuzz testing可以使运行于UNIX系统上的至少25%的程序崩溃; 2002年Aitel^[2]通过自己设计实现的Fuzz工具SPIKE成功地发现了多个未知漏洞; 2008年Godefroid等人^[3]利用Fuzz工具SAGE发现大型Windows应用程序中二十多个未知漏洞。可见, Fuzz测试在安全领域的重要性。

1 Fuzz技术的原理

Fuzz技术的原理简单, 基本的思想是将随机数据作为程序的输入, 并监视程序执行过程中产生的任何异常, 记录下导致异常的输入数据, 从而定位软件中缺陷的位置。因此, 早期的Fuzz技术仅仅是一种简单的随机测试技术^[4], 但却能有效地发现许多程序中的错误。然而, Fuzz技术不可避免地带有随机测试产生的测试时间长, 大量冗余测试输入, 覆盖率低等

基金项目: 国家自然科学基金(编号: 60873191, 60903152, 61003286, 60821001)

作者简介: 赵丽娟(1986), 女, 学生, 密码学网络安全. E-mail: zhaolj1921@163.com

缺点^[5]，并且早期的Fuzz测试只能测试程序的初始状态，而很多程序特别是网络协议程序的很多错误是隐藏在程序的后序状态中的。

基于网络协议的 Fuzz 利用“畸形数据包”测试网络环境的健壮性。一个基于网络协议的 Fuzz 测试的实现过程如下：

- ① 获得待测协议的正常数据包
- ② 用变异数据替换该数据包中的某些部分
- ③ 用发包器向目标应用发包
- ④ 观察目标应用的反应

通常情况下，通过抓包器捕获客户端与被测设备正常交互的数据包作为测试的正常数据包样本。通过任意方式改变随机数据。例如，可以打乱整个数据包，也可以把数据包中的某个部分替换。不管采用什么方法变异数据，关键是在数据包中放入大量随机数据，然后将该数据包发送到目标应用并观察目标应用的行为能力。

2 Fuzz 测试工具—Peach

Peach是一款非常著名的，采用Python语言编写的跨平台的模糊测试框架，它最初由IOACTIVE于2004年发布^[6]。

Peach 的测试对象几乎包括了所有常见的 Fuzz 对象，例如文件结构，com，网络协议，API 等。

Peach把用于数据定义的文件叫做Peach pit file。使用Peach时，实际上主要工作就是定义这样一个xml文件指示Peach测试平台去做测试。Peach pit file基本上总是包含以下几个部分^[7]：

```
<?xml...版本，编码之类...>
<Peach ...版本，作者介绍之类...>
<Include ...注意有两个文件是一定要被包含进来的/>
<DataModel >
    原始数据结构定义
</DataModel>
<StateModel >
    测试逻辑，状态转换定义，如收到什么样的数据包之后，发出什么样对应的数据包
</StateModel>
<Agent >
    检测 exception，crash 等
</Agent>
<Test >
    指定将要使用到的 state，agent，publisher 等
```

</Test>

<Run >

Fuzzer 执行的进入点

</Run>

</Peach>

- 1) 整个文件被一个大标签<Peach> </Peach>包括。
 - 2) 文件中的第二级标签包括 Include, DataModel, StateModel, Agent, Test, Run 共 6 种。
 - 3) Include 包含的外部文件, 其中 defaults.xml 和 PeachTypes.xml 是必须的, 里边含有 Peach 的基本方法, 类, 数据类型等。
 - 4) DataModel 用于定义数据结构, 此标签下还可以有若干级、若干种下级标签。使用这些子标签可以比较容易的定义数据的类型, 大小, 各个数据块之间的关系, 以及 CRC 校验和等。还可以定义多个 DataModel, 多个 DataModel 之间可以有关系也可以没有关系。
 - 5) StateModel 用于定义测试的逻辑, 实际上相当于一个状态机。下级标签包括 State, 每个 State 中又可以包含若干个 Action 标签。State 表示一个状态, 不同的 State 之间可以根据一些判断条件进行跳转。Action 用于执行打开文件, 发送数据包之类的命令。
 - 6) Agent 是一个主要功能是用来监测被测目标的反应, 如 crash 等。
 - 7) Test 这个标签域比较简单, 一般只是制定使用哪个 Agent, 哪个 StateModel, 用什么方法发数据, 有时还会指定使用什么方法加工(变异)数据。
 - 8) Run 这个标签域也比较简单, 指定当前这次 Fuzz 测试使用哪个 Test。
- 图 1 是一个 helloworld 的例子, 了解 Peach pit file 的基本结构和 Peach 运行的框架。

```
<?xml version="1.0" encoding="utf-8"?>
<Peach xmlns="http://phed.org/2008/Peach" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://phed.org/2008/Peach ../peach.xsd" version="1.0"
  author="Michael Eddington" description="Hello World Example">
  <Include ns="default" src="file:defaults.xml" />
  <DataModel name="HelloWorldTemplate">
    <String value="Hello World!" />
  </DataModel>
  <StateModel name="State" initialState="State1">
    <State name="State1">
      <Action type="output">
        <DataModel ref="HelloWorldTemplate"/>
      </Action>
    </State>
  </StateModel>
  <Test name="HelloWorldTest">
    <StateModel ref="State"/>
    <Publisher class="Stdout.Stdout" />
  </Test>
  <Run name="DefaultRun" description="Stdout HelloWorld Run">
    <Test ref="HelloWorldTest" />
  </Run>
</Peach>
```

图 1 helloworld 示例

Fig.1 The sample of helloworld

我们要做的就是运行下这个 Peach pit file, 在命令行下输入该命令, C:\Peach\Peach.py C:\Peach\samples\HelloWorld.xml

我们可以看到屏幕上一开始打印“Hello World!”, 之后 Peach 会以这个原始的字符串为模板变异出许多畸形的数据出来, 包括了超长串, NULL 结束符缺失的非法串, 格式化串等有可能引起程序出错的串, 然后依次打印出来。图 2 给出了程序执行的效果图。

```
G:\peach>peach /samples/HelloWorld.xml
! Peach 2.3.6 Runtime
! Copyright (c) Michael Eddington

Warning: Run 'DefaultRun' does not have logging configured!
[1] Starting run "DefaultRun"
[1] Test: "HelloWorldTest" <None>
[1:?:?] Element: N/A
Mutator: N/A

Hello World!
[2:4397:?] Element: N/A
Mutator: DataTreeRemoveMutator

[3:4397:?] Element: N/A
Mutator: UnicodeUtf8ThreeCharMutator

[4:4397:?] Element: N/A
Mutator: DWORDSliderMutator

o World!
[5:4397:?] Element: N/A
Mutator: StringMutator

Peach
```

图2 helloworld 执行效果图

Fig.2 The result of run helloworld

3 HTTP 协议

HTTP (HyperText Transfer Protocol) 是超文本传输协议, 是客户端浏览器或其他程序与Web服务器之间的应用层通信协议^[8]。HTTP协议采用请求/响应模型。客户端向服务器发送一个请求, 请求头包括请求的方法、URL、协议版本、以及包含请求修饰符、客户信息和内容的类似于MIME的消息结构。服务器以一个状态行作为响应, 相应的内容包括消息协议的版本, 成功或者错误编码加上包含服务器信息、实体元信息以及可能的实体内容。

一次HTTP的工作过程可以分为以下四步^[9]：

- 1) 首先建立客户机与服务器的连接。如在浏览器的地址栏处输入访问地址或者点击某个超链接等。
- 2) 建立连接后, 客户机给服务器发送一个请求。请求方式的格式为: 统一资源标识符 (URL)、协议版本号, 后边是MIME信息包括请求修饰符、客户机信息和可能的内容。
- 3) 服务器接到请求后, 给予相应的响应信息。其格式为一个状态行, 包括信息的协议版本号、一个成功或错误的代码, 后边是MIME信息包括服务器信息、实体信息和可能的内容。
- 4) 客户端接收服务器返回的信息, 显示在客户端的浏览器显示屏上, 然后客户机与服务器断开连接。

4 Peach 测试 HTTP 协议的设计与实现

某设备提供 Web 服务, 该服务正常情况下通过在浏览器的地址栏里输入网站的地址,

提供正确的用户名和密码后正常登陆。本实验主要目标是测试 HTTP 协议的健壮性,通过定制 Peach pit file,查看是否可以通过溢出登录网址等方法导致该服务不响应或者可以绕过口令登录等异常现象。

本实验通过 Peach 对 HTTP 协议进行测试的方法与传统 Fuzz 测试不同,并不是测试全部字段,而是只测试通过协议分析到的可能发现漏洞的感兴趣字段。这种方法不仅可以找到漏洞,而且很大程度上提高了测试效率。图 3 为 Peach pit file 中 HTTP 协议的数据定义部分,并且在测试过程中通过使用<Include xpath="...">选择我们经过分析后想要测试的字段。

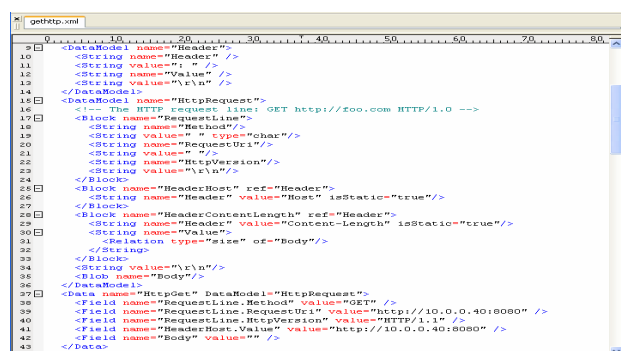


图 3 Peach 测试 HTTP 协议的数据定义部分代码

Fig.3 The code of DataModel for http-based Peach testing

下面是对某设备提供的Web服务进行HTTP协议健壮性测试的主要效果图。在执行程序的同时,用Wireshark^[10]监测被测设备反应,如图 4,图 5,图 6 所示。刚开始发包的时候可以正常建立TCP连接,然后发送HTTP畸形数据包。当发送完某个畸形数据包后,就无法再建立TCP连接,服务器端没有响应TCP连接。直接在浏览器地址栏输入访问地址也无法打开主页面。只有重新启动该服务才能进行访问。

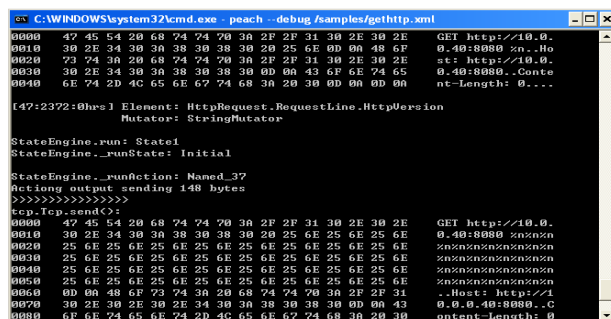


图 4 Peach 控制台下 debug 模式执行效果图

Fig.4 The debug console result for http-based Peach testing

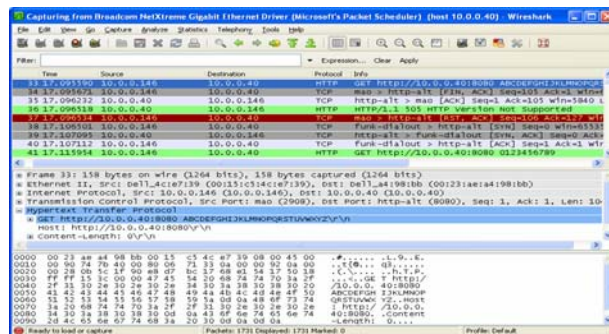


图 5 Peach 测试 HTTP 协议效果图 1

Fig.5 The wireshark result for http-based Peach testing 1

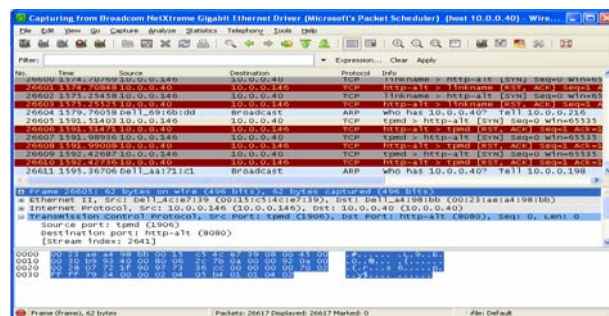


图 6 Peach 测试 HTTP 协议效果图 2

Fig.6 The wireshark result for http-based Peach testing 2

可见，在用 **Peach** 测试到某个畸形数据包后，无法再建立 **TCP** 连接，并且控制台中正在执行的 **Peach** 也因为没有等到 **TCP** 连接响应而一直处于等待状态。在 **debug** 模式下执行 **Peach**，可以首先将控制台上等待处的畸形数据作为漏洞触发对象。重启 **Web** 服务后，利用该畸形数据组装成数据包，用 **Peach** 简单的将这一个包发送给被测设备，发现被测设备提供的 **Web** 服务并没有因为这一个畸形数据包而产生异常。说明这一个畸形包并不能触发该漏洞。为了找到漏洞，重新执行一遍 **Peach** 测试，发现同样是发送完该数据包后服务器无法正常连接，那么此时考虑该漏洞不是因为一个畸形包触发的，很可能是一组畸形包导致的。于是利用 **Peach** 运行时提供的 **range** 参数，每次执行一部分测试用例，看是否导致服务器异常，如果导致异常则进一步缩小范围进行测试，重复这些过程，直到尽可能小的确定触发该漏洞的范围。最后发现，正如我们所推测的一样，导致服务器无法响应客户端的是一组畸形包。

5 小结

本文利用基于框架的 Fuzz 测试工具 Peach，针对 HTTP 协议设计 Peach pit file，实现了对某设备 Web 服务的 HTTP 协议的安全性测试。为了提高测试的效率，对于协议健壮性的测试可以只测试我们感兴趣的有特殊意义的字段，而不需要把每个字段都进行测试。这样可以避免做一些无用功，可以在有限的时间内完成最有效的测试。因此，及早展开有效的协议安全测试有助于最大限度的避免来自于网络应用的危害。

[参考文献] (References)

- [1] MILLER B P, FREDRIKSON L, SO B. An empirical study of the reliability of UNIX utilities[J].
Communications of the ACM, 1990, 33(2):32.
- [2] AITEL D. The advantages of block-based protocol analysis for security testing[R]. New York: Immunity Inc, 2002.
- [3] GODEFROID P, LEVIN M, MOLNAR D. Active property checking[C]//Proc of the 8th ACM International Conference on Embedding Software. 2008:19-24.
- [4] MILLER B P, KOSKI D, LEE C P, et al. Fuzzing revisited: a reexamination of the reliability of UNIX utilities and services[R]. Madison: University of Wisconsin Madison, 1995.
- [5] OFFUTT A J, HAYES J H. A semantic model of program faults[C]//Proc of ACM SIGSOFT International Symposium on Software Testing and Analysis. San Diego: ACM Press, 1996.
- [6] Peach[EB/OL]. (2009-06). <http://www.peachFuzzer.com>; <http://peachfuzz.sourceforge.net>.
httpfuzz/googlegroups.com.
- [6] Ircfuzz[EB/OL]. (2009-06). <http://www.digitaldwarf.be/products/ircfuzz.c>.
- [7] <http://apps.hi.baidu.com/share/detail/16178380>.
- [8] RFC 2616 - Hypertext Transfer Protocol. <http://www.faqs.org/rfcs/rfc2616.html>.
- [9] 刘远生, 辛一, 薛庆水. 计算机网络安全. 清华大学出版社, 2006.
- [10] Wireshark [EB/OL]. (2009-06). <http://www.wireshark.org>.