FirmFuzz: Automated IoT Firmware Introspection and Analysis

（物联网固件自动化自省和分析- S&P 2019）

斯琴 2020.11.13

论文简介

本文提出 FirmFuzz，一个独立于嵌入式设备的自动化仿真和动态分析框架，针对基于 linux 的固件镜像，采用了一种基于灰盒的 fuzzing，结合静态分析和系统自省，为固件镜像提供有针对性和确定性的漏洞发现。

应用场景：针对单一固件，深入分析检测漏洞。

结果：评估了来自 27 个不同的设备的 32 个镜像，在网络可达环境下，发现了 6 个不同设备的 7 个未公开漏洞 + 4 CVE 。

优势：本文重点侧重于分析深度（测试触发各种各样的漏洞的代码路径深度），而不是广度（一次能分析很多固件）。

工作流程：

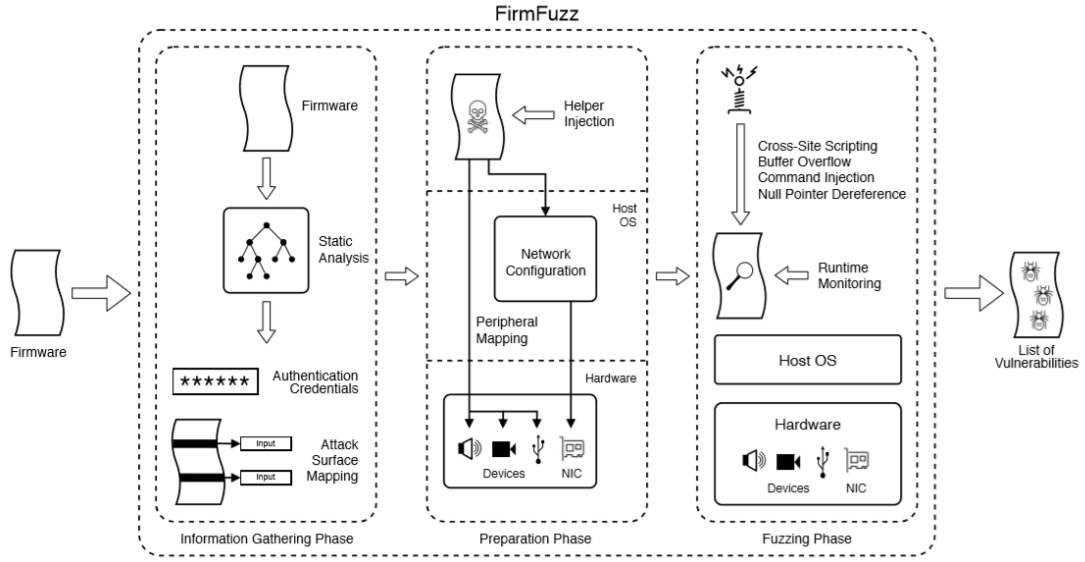


Figure 1: FirmFuzz Workflow

整个框架工作流程分为三个阶段，信息收集阶段，准备阶段，fuzzing 阶段。

信息收集阶段，主要实现两个目标：①使用预置语料库，对 web 登陆界面进行暴力破解，对固件进行强制身份验证。②静态分析攻击面，寻找 php 中易受攻击的代码输入。

准备阶段，配置仿真环境，通过修改内核和文件系统，实现①外设映射，将固件所需的设备映射到假设备驱动程序，在查询时总返回 True。②文件系统中注入 helper 二进制文件，辅助漏洞检测。③ 推断网络配置，创建虚拟网络接口。

fuzzing 阶段，运行自己搭建的 fuzzer 框架，fuzzer 分为两部分，分别运行在主机端（外部）和客户机端（固件内部）。当固件仿真成功运行起来之后，固件将作为一个服务器，

提供 web 服务。

① 处于主机端的 fuzzer 首先使用无头浏览器通过 web application 与固件通信，其通信产生的网络流量被 proxy server 拦截，并分析推断出合理 http 请求，作为种子输入。

② 处于主机端的 fuzzer 中的 payload 模块，根据攻击目标的漏洞，修改合法请求，并把他传递给固件。

③ 处于客户机端（固件内部）的 fuzzer 模块，实施运行时监控，结合自省技术，和之前注入文件系统的辅助二进制文件， 通过分析日志，检测是否触发漏洞。

④ 如果检测到漏洞，就保留修改的请求作为触发漏洞的 PoC（概念验证），同时记好接收方的 url, 方便重现。

本 fuzzer 框架的特点和优势：① 上下文驱动产生推断输入生成 ②实现了确定性漏洞检测 ③ 消除了 fuzz ing 过程中的不一致性

问题解答

1. 什么是无头浏览器（headless broswer）？

无头浏览器，指的是没有图形交互界面 GUI 的浏览器。去掉 GUI 是因为 Headless 模式下的 Chrome 开放了编程接口，可以通过代码控制它的行为，而不再需要图形界面，也就是所谓的浏览器自动化。相对于一般浏览器来说，节省了 GUI 所必须消耗的大量内存，给多线程、进程并行提供了方便。

无头浏览器在类似于流行网络浏览器的环境中提供对网页的自动控制，但是通过命令行界面或使用网络通信来执行。它们对于测试网页特别有用，因为它们能够像浏览器一样呈现和理解超文本标记语言，包括页面布局、颜色、字体选择以及 JavaScript 和 AJAX 的执行等样式元素，这些元素在使用其他测试方法时通常是不可用的。

无头浏览器可以部署在服务器端，根据事先写好的代码对客户端请求的页面做预渲染，将渲染好的静态页面数据发送给客户端，这样可以省去客户端页面首次渲染所需要的时间，让页面生成的速度显著提高。

应用场景：

无头浏览器通常用来：

- Web 应用程序中的自动化测试
- 对 JavaScript 库运行自动化测试
- 拍摄网页截图
- 收集网站数据
- 自动化网页交互

除了以上提到的测试和截屏，无头浏览器还可以被用来自动执行恶意任务。最常见的形式是做网络爬虫，或伪装访问量，或探测网站漏洞。

一个非常流行的无头浏览器是 PhantomJS，因为它是基于 Qt 框架，所以跟我们常见

的浏览器相比有很多不同的特征，因此有很多方法判断出它。但是，从 chrome 59 开始，谷歌发布了一款无头谷歌浏览器。它跟 PhantomJS 不同，它是基于正统的谷歌浏览器开发出来的，不是基于其它的框架，这让程序很难区分出它是正常浏览器还是无头浏览器。

2. 来自哪个团队?

Purdue University(普渡大学)

3. 整个框架是否基于 Firmadyne？固件内核，文件系统是否经过修改?

整个仿真框架不是基于 firmadyne，而是自己新构建的，固件的内核和文件系统都经过修改，再基于 qemu 进行仿真。

修改的内容:

内核：采用外设映射策略，创建了一个极其相似于固件预期的运行环境。(镜像使用自定义的内核进行仿真，内核配置为在访问不支持的设备时 panic，firmfuzz 使用这个 panic 日志，创建一个设备到假设备驱动的映射，迭代执行这个过程，直到所有不支持的设备被映射到一个假设备)。

文件系统：利用全系统仿真，向文件系统中注入 helper 二进制文件，检测 CI 漏洞。

4. 什么是确定性发现漏洞

确定性发现漏洞，就是指保证检测到的漏洞是一定存在而且可复现的。firmfuzz 框架修改了仿真运行环境和固件本身。通过监测当测试输入执行后，经过修改扩展的固件产生的 log 日志来检测是否触发漏洞。

现有检测漏洞的方法是通过漏洞扫描器（例如 ZAP），由于他们盲目的依赖服务器反馈，会产生很高的假阴性（FN），而且服务器端的 response 可能也不准确，再加上缺少一些 Linux utility，有些漏洞可能不能本地化复现，或者错过一些漏洞。

本 fuzzer 框架对被测固件实行运行时监控，以及上下文驱动的输入生成，通过监视 helper 二进制文件和内存访问冲突处理程序的执行，确保了由 firmfuzz 捕获的属于 CI，BO，和 NPD 类 bug 都不是误报，而且还提供了哪些固件资源存在 bug 类信息，是对现有扫描器的重大改进，现有的扫描器由于缺少系统自省功能，既不能在存在 bug 时本地化复现漏洞，也不能保证这些扫描器检测到的 bug 不是误报，都需要进一步手工分析。

5. 判断漏洞是否通过分析日志?

是，通过分析日志。

当执行测试输入时，通过监视被扩展增强的固件生成的日志， Firmfuzz 检测 CI, BO, NPD, XSS 四类漏洞。

对于 CI：监视 execve 系统调用，以记录 helper 二进制文件的执行情况，

对于 BO，NPD：监视内核日志，看看是否有固件进程试图访问未映射的内存

对于 XSS： 不需要客户端协助，由 Firmfuzz 进行的主机端监控足以检测，监视形式主要是检测作为 payload 发送的 javascript 代码片段是否在机器中执行

6. fuzzing 过程及如何发现漏洞?

使用 Qemu2.5.0 作为仿真后端，使用 Selenium Webdriver 3.4.0 无头浏览器作为 fuzzing 的驱动之一，使用 mitmproxy 0.18.2 来监控 fuzzer 和仿真固件之间发送的网络流量。

使用自定义的自动化生成型 fuzzer。

三个特点：上下文驱动 input 生成，确定性漏洞检测，消除 fuzz 过程中的不一致性

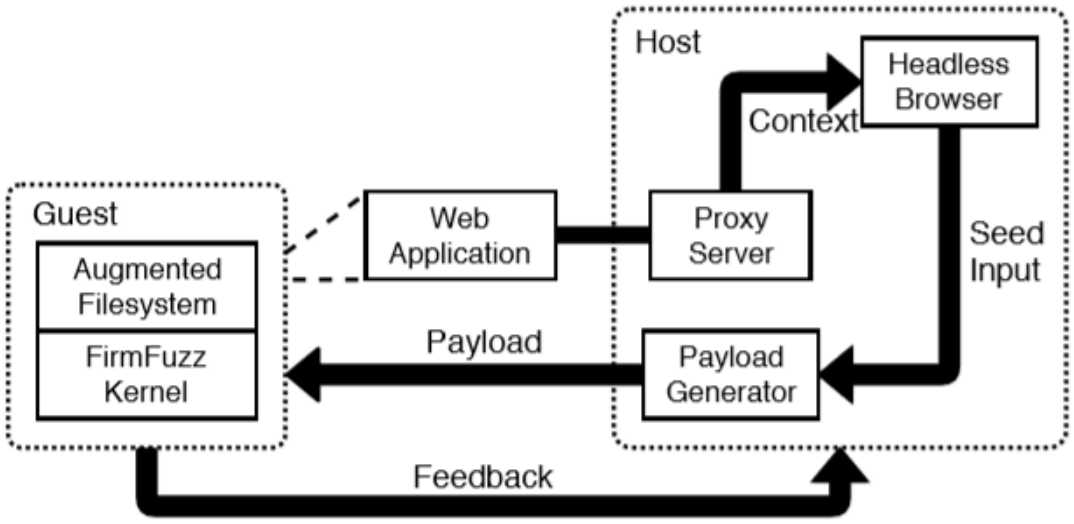


Figure 2: Fuzzing Workflow

本框架的 fuzzer 需要一系列 web 应用程序可接受的种子输入和有效 url，由两部分构成，在 host 端推断合法 input 作为种子，及种子变异，传递 payload，在 guest 端监控是否触发漏洞，并记录反馈。

固件运行的 Web 应用程序由客户端（网页显示）javascript 代码和服务器端代码组成。Firmfuzz 利用无头浏览器，通过 web application 和固件交互，执行客户端（网页）javascript 代码。使用交互过程中的上下文信息，产生 fuzzing input。

所有的 fuzzer 和固件之间的网络流量经过 proxy server，使得 proxy server 可以捕获能被 firmfuzz 变异的候选输入。

在交互过程中，为了与 web 程序应用接口进行交互，firmfuzz 需要了解固件的 web 界面设置，由于不同供应商之间的 web 界面有很大差异，所以 firmfuzz 为不同的 web 接口创建了模板，需要一些手工分析，这也是将分析扩展应用到其他设备的一个限制因素。

7. 如何解决 web 界面登陆问题?

通过认证凭据语料库，采用暴力破解方式对固件进行强制身份验证

8. Evaluation 中数量下降原因

Table 1: Firmware images tested

Vendor	Scraped Images	Linux FS found	Network Inferred	Fuzzed (Unique Devices)	Unique Web UI
TRENDnet	359	129	26	6 (5)	2
Netgear	2646	675	162	20 (17)	3
D-Link	3422	209	15	6 (5)	1
Total	6,427	1,013	203	32 (27)	6

203-32 存在一个锐减，因为在仿真过程中，缺少一些对特殊需求的设备的仿真（例如，有些系统在 web 服务器启动之前需要先获取到摄像头，否则无法启动），如果没有获取到 web 接口作为 fuzz 入口点，无法开展 fuzzing 分析。

6-在同一厂商的不同设备中，web 接口有很高的可重用性，例如 27 个来自 3 个不同厂商的设备，一共可分为 6 种不同的 web UI 接口，建立好模板后，即可开展 fuzzing 测试。

9. 本文与 firmadyne 进行对比

	Firmfuzz	Firmadyne
分析范围	深度，针对单一固件，为各种各样的漏洞，测试深度代码路径 重点在于发现单一固件的 web 漏洞	广度，采用通用的方法，批量仿真和分析固件 重点在于通用性的固件批量仿真平台
网络配置	在 network infer 模式下运行仿真，记录固件和内核网络接口之间的交互，使用这些日志记录，推断网络配置，并创建适当的虚拟网络接口。	
外设映射	给请求的未知外设提供一个假的外设驱动程序，在被查询时总是返回 True	无外设映射策略，但创新点在于仿真了 NVRAM
发现漏洞	自制 fuzzer 框架，根据目标固件采用上下文驱动的方法，定制 payload，测试深层代码路径，自动化发现未知漏洞	运行 metasploit 和自己的 PoC 验证手工发现的已知漏洞
检测漏洞	CI：helper 二进制文件注入，一旦 helper 被执行，标记发现 CI。完全自动化，与固件无关	CI：不支持 CI 漏洞检测
	BO：类似于 firmadyne 的异常处理机制，但是不需要手动分析，用自定义的 fuzzing 驱动程序，可自动触发仿真固件中的 BO 漏洞，并打	BO：手工发现特定 netgear 路由器上的 BO 漏洞。通过向特定网页发送 curl 请求来触发该漏洞

	包 PoC 用于重现漏洞。	
	NPD：异常处理机制	
	XSS：从 host 端监控来检测	

10. 思考

我认为本文 fuzzing 框架设计的很好，检测部分用到了自省监控，fuzzing 部分针对 web 界面存在的按钮和一些可输入点，先推断合法输入，再变异传递 payload，试图触发漏洞。虽然变异只是简单的文本替换，但根据他最终的实验结果和对比可以证明他这套方法是有效的。不足之处可能在于，最终能够 fuzz 的设备数量稍微有点少，而且针对 web 界面 fuzz 需要模板，所以不能继续大规模扩展。

拓展知识

1. 嵌入式系统构成

主要由 bootloader、kernel、以及根文件系统三部分组成。在 flash 中分配了存放内核、根文件系统的区域。bootloader 加载了内核，内核启动，加载文件系统，进入 Linux 系统。

2. Linux 内核和文件系统的关系

1）文件系统

文件系统指文件存在的物理空间，linux 系统中每个分区都是一个文件系统，都有自己的目录层次结构。Linux 文件系统中的文件是数据的集合，文件系统不仅包含着文件中的数据而且还有文件系统的结构，所有 Linux 用户和程序看到的文件、目录、软连接及文件保护信息等都存储在其中。这种机制有利于用户和操作系统的交互。在 Linux 文件系统中，EXT2 文件系统、虚拟文件系统、/proc 文件系统是三个具有代表性的文件系统。

2）根文件系统

根文件系统是内核启动时所挂载的第一个文件系统，内核代码的映像文件保存在根文件系统中，系统引导启动程序会在根文件系统挂载之后从中把一些初始化脚本（如 rcS，inittab）和服务加载到内存中去运行。

3）内核

内核是操作系统的核心，负责管理系统进程、内存、设备驱动程序、文件和网络系统，决定系统的性能和稳定性。主要模块分为以下几个部分：存储管理、CPU 和进程管理、文件系统、设备管理和驱动、网络通信，以及系统的初始化（引导）、系统调用等。

操作系统是用来和硬件打交道，并为用户程序提供一个有限服务集的低级支撑软件，计算机是一个软

件硬件的共生体，完成控制硬件工作的软件称为操作系统。

4）内核和文件系统的关系

内核指的是一个提供硬件抽象层、磁盘及文件系统控制、多任务等功能的系统软件。

文件系统是内核的一部分。文件系统实现了系统上存储介质和其他资源的交互。kernel tree 中的 fs 目录都是关于文件系统的，可以说文件系统是内核的一个大子系统。

3. 什么是设备驱动程序

设备驱动程序用来将硬件本身的功能告诉操作系统，完成硬件设备电子信号与操作系统及软件的高级编程语言之间的互相翻译。设备驱动程序通常会占到 70%以上份额的操作系统内核源码，且设备驱动程序的更新维护往往会牵涉到超过 35%的源码修改，故而保持设备驱动程序与操作系统内核不断变化的其余部分的一致性是操作系统内核开发的一项难题。

4. OWASP ZAP

全称 OWASP Zed Attack Proxy，Zed 攻击代理服务器是世界上最受欢迎的免费安全工具之一。ZAP 可以帮助我们在开发和测试应用程序过程中，自动发现 Web 应用程序中的安全漏洞。另外，它也是一款提供给具备丰富经验的渗透测试人员进行人工安全测试的优秀工具。

在安全性测试领域，安全性测试主要可以由以下几种测试策略来覆盖：

- 漏洞分析 - 对系统进行扫描来发现其安全性隐患
- 渗透测试 - 对系统进行模拟攻击和分析来确定其安全性漏洞
- 运行时测试 - 终端用户对系统进行分析和安全性测试（手工安全性测试分析）
- 代码审计 - 通过代码审计分析评估安全性风险（静态测试，评审）

ZAP 主要是用于上述的第二种测试，即渗透性测试。

ZAP 以架设**代理**的形式来实现渗透性测试，类似于 fiddler 抓包机制。他将自己置于用户浏览器和服务器中间，充当一个中间人的角色，浏览器所有与服务器的交互都要经过 ZAP，这样 ZAP 就可以获得所有这些交互的信息，并且可以对他们进行分析、扫描，甚至是改包再发送。

5. Web Application 简介

常见 web 架构可分为客户端/服务器（C/S 架构）和浏览器/服务器工作（B/S 架构）

1）C/S 架构

将单机连成网络，使它们之间提供服务，如 A 向 B 提供服务。常见的服务是文件共享，FTP 文件下载等。把提供（响应）服务的计算机称作服务器（Server），接受（请求）服务的计算机称作客户机（Client）

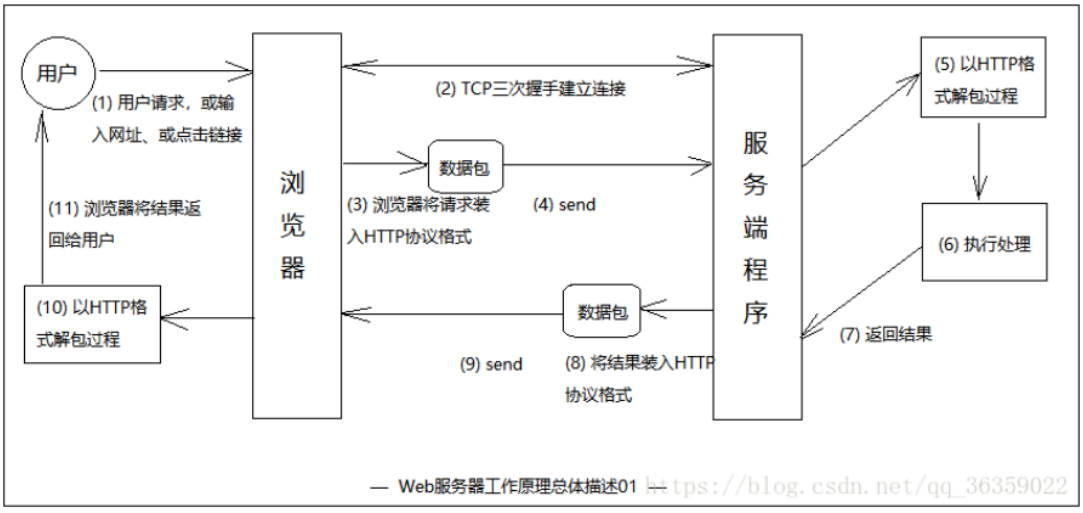
C/S 架构程序分为两部分：服务器端和客户机端，分别称为服务器端程序（或服务程序）和客户端程序（或客户程序）。对于客户端程序，对每一个客户机，也都要分别安装。安装好了客户端程序后，就可以通过通信线路与服务器交互，或通过服务器，与其他客户机通信。（例如 QQ）

2）B/S 架构

若通过客户机中的浏览器(Browser)，向服务器发出请求，接收其响应的结果，这样的协作方式为 B/S 方式，或 B/S 架构。

此时，客户端程序就是浏览器，而浏览器的安装是随着操作系统的安装完成的，不需要用户额外安装。对用户而言，使用 B/S 程序（如上网看新闻，收发电子邮件），不需要安装专门的客户端程序，直接在浏览器中操作即可。这使得 B/S 程序的维护十分方便，因为不用管客户端程序，只要维护好服务器端程序即可。

3）服务器工作原理



简单而言，作为一个服务器，其根本工作分为：1.接收数据 2. 发送数据 3. 数据处理

4）客户端工作原理（B/S 架构）

任何应用系统都必须有一个提供用户操作的界面，即用户界面。浏览器的工作，从整个 B/S 程序来看，是用户与 B/S 程序打交道的一个界面（接口）。它的任务是：

A.收集用户输入的数据

B.将用户数据发送到服务器

C.接收服务器返回的响应

D.解释，执行这些代码

因此，浏览器扮演的是服务器在用户那里的一个代理（Agent）的角色。这个代理，具有收集消息，请求响应和解释服务器发回的指示的作用。

参考链接：

https://blog.csdn.net/datase/article/details/78145754 内核和文件系统

https://www.jb51.net/article/58978.htm web 程序工作原理详解

https://blog.csdn.net/qq_36359022/article/details/81666221 web 服务器工作原理详解