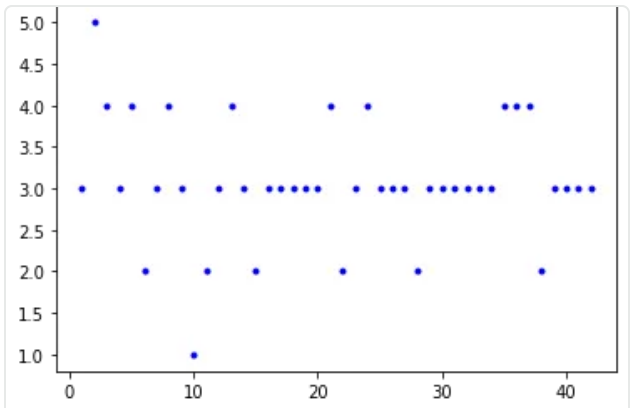
分析结果

PATH图应该不是结果，但是放在了项目中“path图”文件夹中！！！

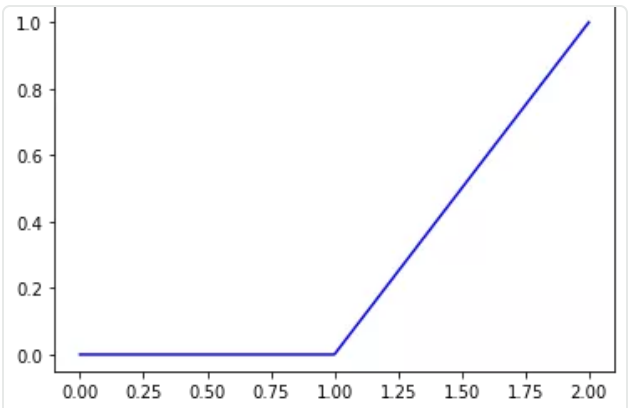
MutationScore结果图（满分均为100分）

> 这些图片的 X轴 为 测试用例（按生成顺序运行 **所以可以看作时间轴**） Y轴为MutationScore 满分均为100，有些（e.g. binutils readelf)其实mutationScore很低
>
> 每个MutationScore都不是整数（只是我们在生成的测试报告json文件中所读取到的是整数，实际上都是小数，但这些数据只有在html中网络上的js中通过计算获得的，难以获取）
>
> 有些图片选择了 点图，有些选择了 线图。

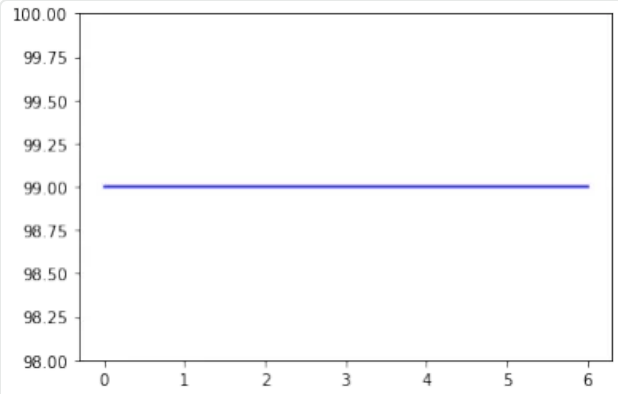
画册视图



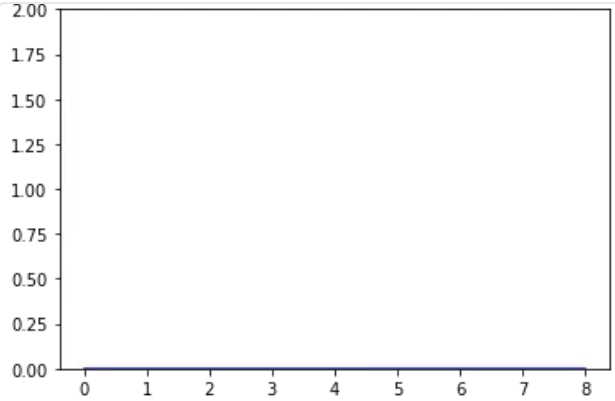
libxml2



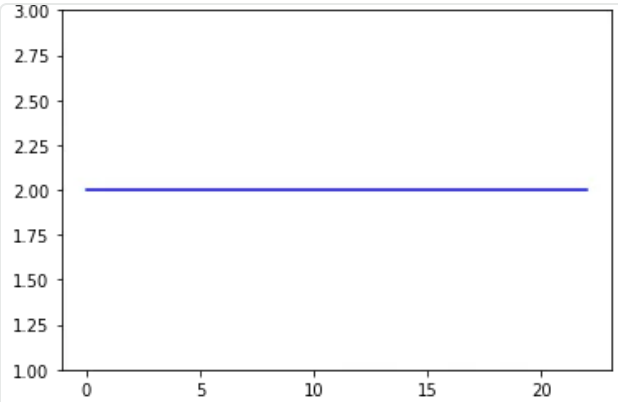
binutils readelf



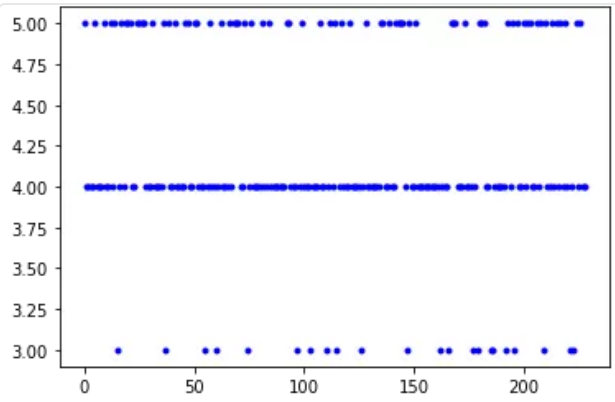
binutils objdump



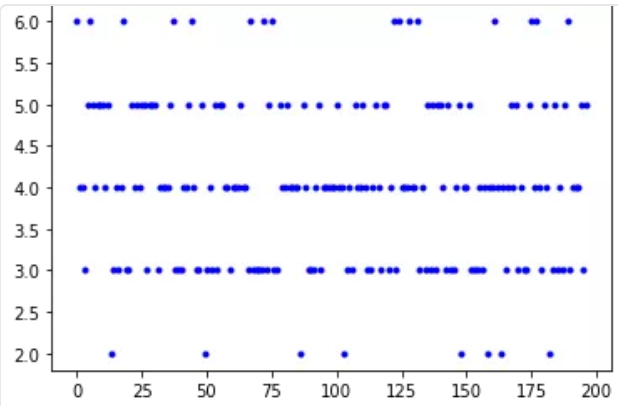
binutils cxxfilt



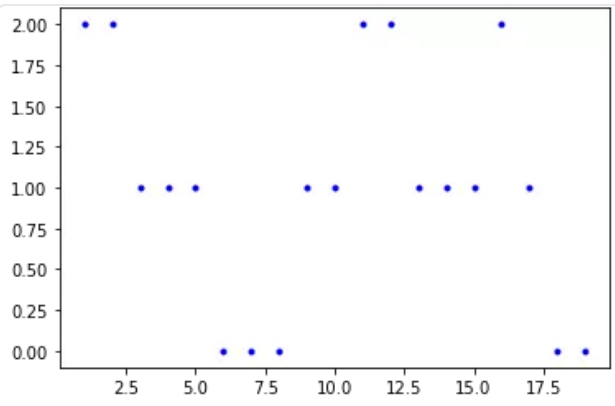
binutils nm



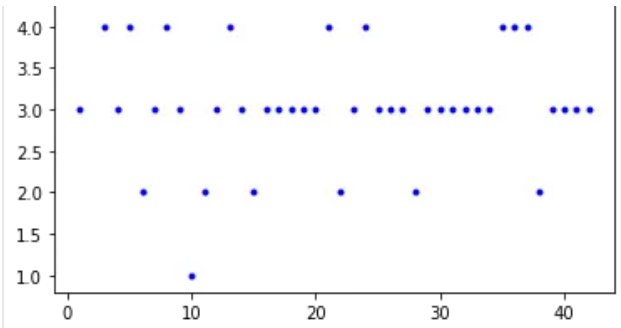
binutils size



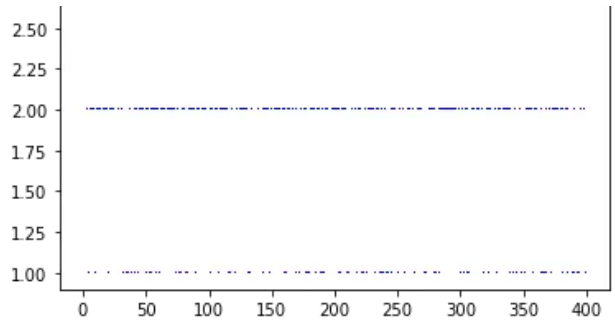
binutils strip



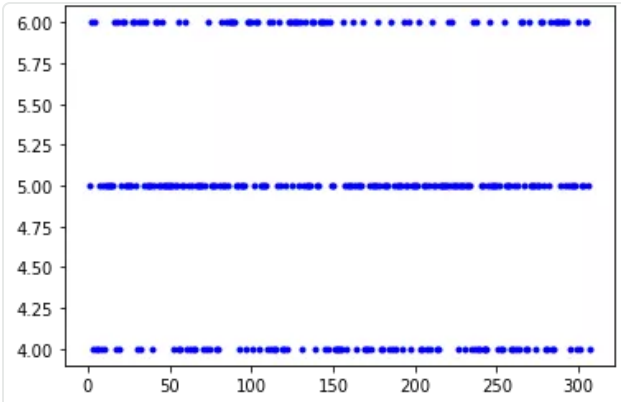
matio



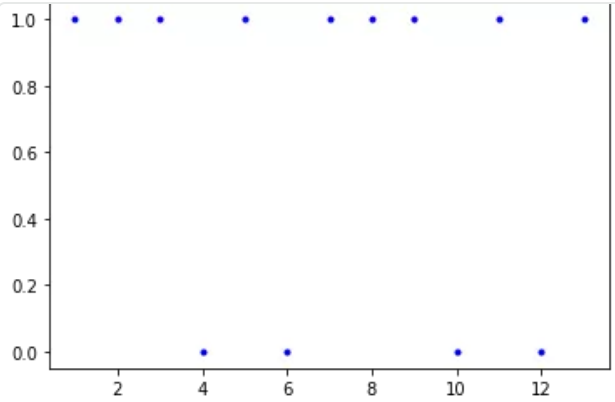
openssl



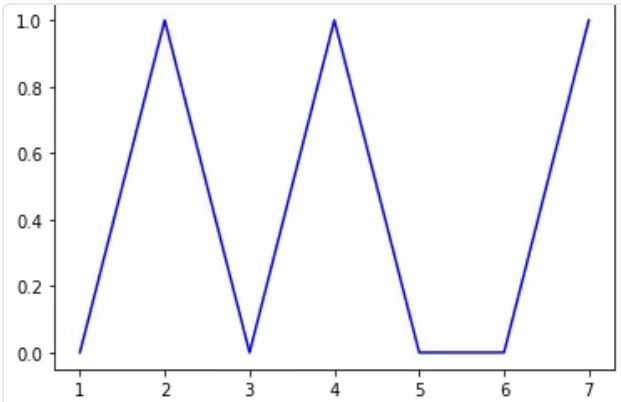
fat



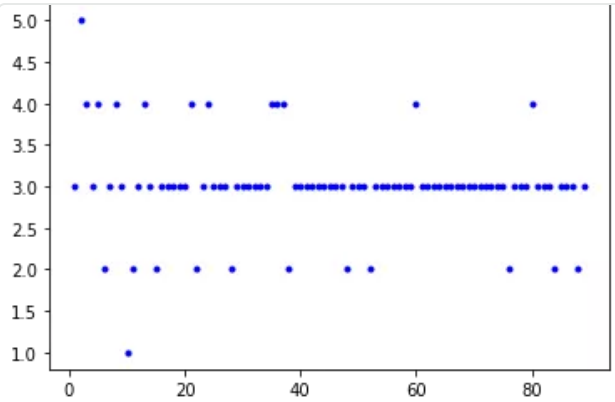
expat



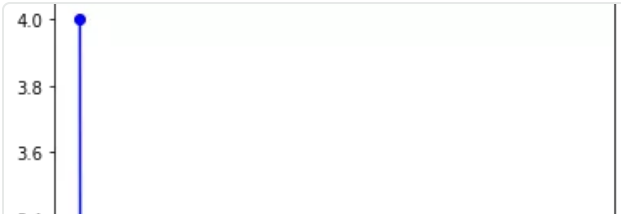
lrzip

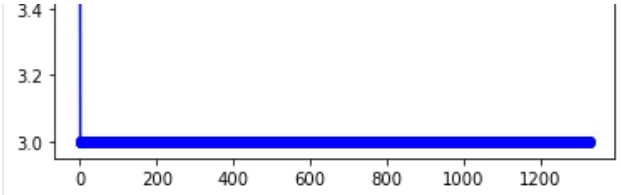


zziplib

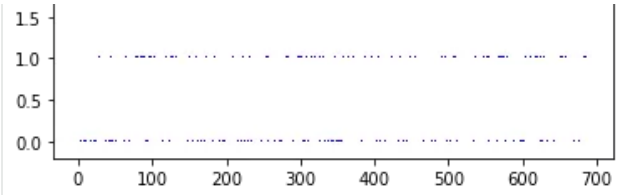


splite

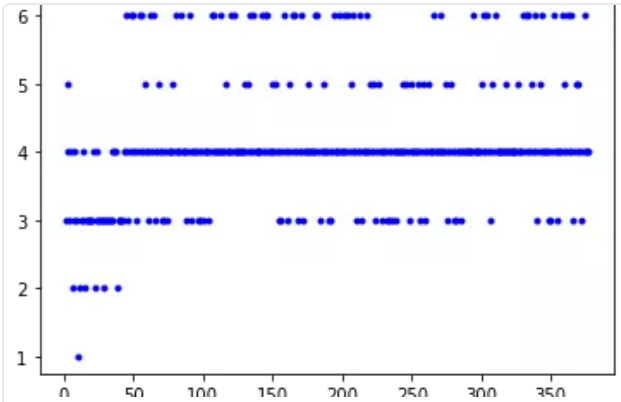




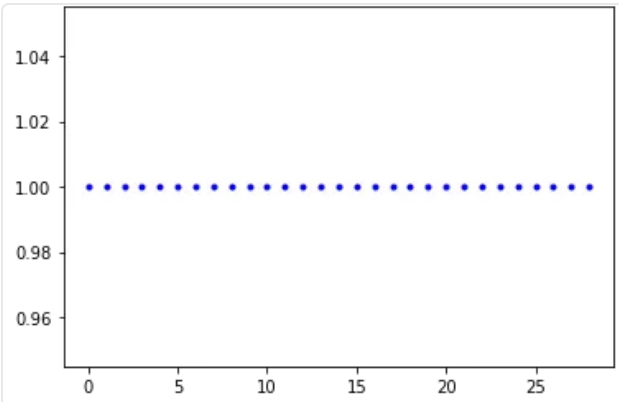
exiv2



xpdf



podofo



w3m

分析

前提

我们每个都使用Mullrunner将生成的测试用例运行完成或者运行超过5小时（有些变异体非常多，而且运行得非常慢，运行不完）。

最后的实验结果是，我们选择的生成种子或许不太好，生成的很多测试用例输入都有着在Original Test中format not recognized的问题，导致会出现很多MutationScore达到99分的数据（这些数据根据我们的判断应该是存在问题的，这些数据都是在Mull中OriginalTest failed的情况下所产生的）。所以在处理这些数据的时候，将99分的所有数据全部删除了，只留下有意义的数据。

然而，运行binutils中的objdump的时候，生成的所有Mutation Score皆为99分，导致我们不得不将这些数据留下，但讨论时避免讨论这个内容。

综上，我们会在上述前提下进行讨论实验数据。

结论

对于AFL生成的测试用例来说，可能是运行时间过短（每个项目都运行了1小时），也有可能是作为种子的测试用例输入质量不佳，生成的测试用例存在很多问题，例如format not recognized 以及 内部文件损坏（无法识别某些文件中内容）。

使用Mullrunner进行运行以后，我们发现MutationScore的分数都非常低，变异杀死率非常低（全部没有超过10%），而且不存在一个稳定上升的趋势。

我们以binutils size为例，所有的Mutation Score集中在4%，有着1%的上下浮动。这意味着对于AFL生成的测试用例，并不能够有效地判断程序中的bug。

以exiv2，只有前几个用例Mutation Score达到了4%，剩下所有的Mutation Score都只有3%。

甚至binutils.cxxfilt中，所有的测试用例都没被杀死，Mutation Score达到了0%。

综上所述，我们认为AFL的模糊在1小时为限制的fuzz中的效果是非常有限的。