[翻译] 使用AFL对CSGO进行模糊测试

Anciety / 2018-09-06 19:33:16 / 浏览数 2885 技术文章 技术文章 顶(0) 踩(0)

使用afl对CS：GO进行模糊测试

在RealWorldCTF2018中有一个非常有意思的题叫"P90 Rush
B"，名字本意是说在Valve的游戏"CS：GO"中的一种极限战术。这个题主要是考挖掘和利用CS：GO游戏服务器中地图文件加载部分的一个漏洞。在CTF期间，我利用了一个

因为这个漏洞影响了CS：GO官方的windows客户端，所以其实是可以有资格被Valve的漏洞赏金程序接受的，其实这个洞是以前一个报告的一个小变种。于是我在CTF之后

于是我得到了不错的报酬，之后我就决定花点时间来找找这个目标里的类似情况，并且在这个过程当中学了点黑盒模糊测试的东西，以前一直没机会去学。这篇文章是用来

我觉得有必要提一下Valve认为我的堆相关漏洞（线性溢出和一些半控制的溢出写）有必要修复，在我没有提供完整的利用程序之前，而利用程序会由于aslr而非常难写。无

请务必记住，在黑盒测试方面我还是个新手，很乐意去学习，所以如果我的一些决策不太好，或者我没找到一些可以让事情简单点的工具的话，请务必告诉我，我会非常感

BSP 文件格式和攻击面

CS：GO当中用到的地图文件格式（也可能是所有使用Source游戏引擎的游戏）叫做BSP，是`binary space partition`（二进制空间分区）的缩写，也就是一种简单快捷的把对象在n维空间中表示的方法。另外，这个格式还不止能够表示3D的信息。BSP文件在服务器和客户端都

从安全研究的角度来讲，我们感兴趣的是客户端和服务器共享的最外层解析代码，大多数在2007年Source游戏引擎源代码泄露中我们都能找到。至少在我看来，代码总体上

模糊测试的基本设置

太长不看版：跟着`https://github.com/niklasb/bspfuzz`复现。

简单的说，我决定对linux服务器二进制文件进行模糊测试，而不是客户端（虽然在linux上也能跑）。对一个命令行程序做模糊测试感觉还是比一整个3D游戏正常一点。这个

我看了一个youtube上的一个教程，主要是教怎么用Hammer去做一张非常简单的地图，但是没想到的是这就已经300k大小了。这巨大的尺寸其实主要是因为里边的模型数

你可以这样把这个地图加到服务器里：

```
$ LD_LIBRARY_PATH=`pwd`/bin ./srcds_linux -game csgo -console -usercon \
     +game_type 0 +game_mode 0 +mapgroup mg_active +map test \
     -nominidumps -nobreakpad
```

这样会把位于`csgo/maps/test.bsp`里的地图加载进去。加载过程大概需要15秒多，所以这肯定不能用来直接进行模糊测试。于是我决定自己用服务器二进制文件用到的共

- `engine.so` - 主要的Source游戏引擎代码（包括BSP解析）
- `dedicated.so` - 专用服务器实现（包括应用程序入口）
- `libtier0.so` - 大概和Steam或者应用程序管理相关

于是这个wrapper主要干了这么几件事：

1. 调用`DedicatedMain`(`srcds_linux`二进制文件也这么干的)来启动一个服务器。
2. 通过将`engine.so`中的`NET_CloseAllSocketspatch`掉，来让他重新跳到`startpoint()`函数来重新获取权限。
3. 调用`forkserver()`函数（这儿一会我们会让AFL来fork）
4. 调用`CModelLoader::GetModelForName`来从制定地图文件名加载地图。
5. 以最快速度退出。

这里需要对`engine.so`和`libtier0.so`打几个patch，用一个python脚本就可以了。wrapper和patch脚本都要根据服务器的版本进行调整，主要是针对偏移量改变。

AFL

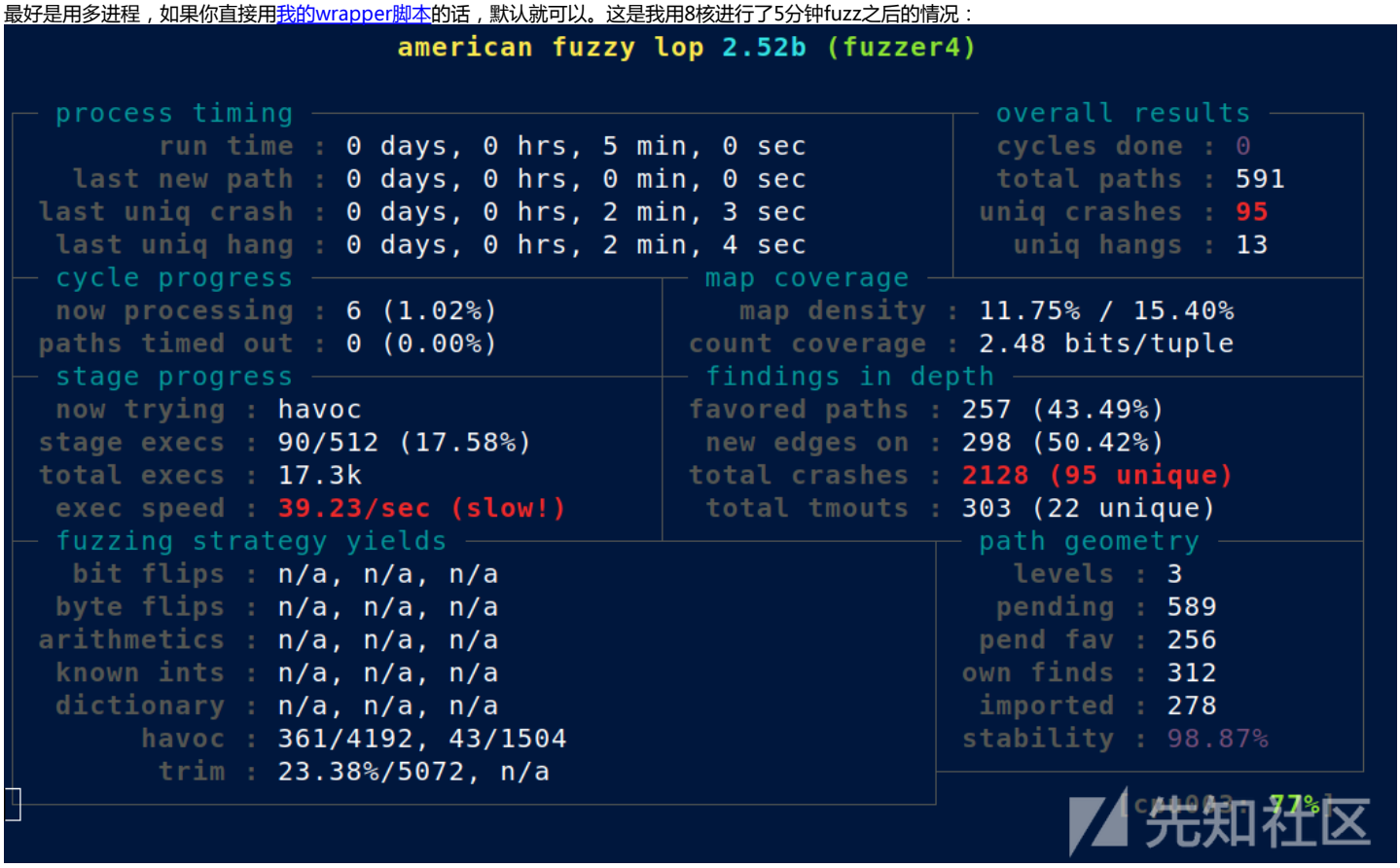
我对AFL做了一点改动:

1. 输入文件必须由`.bsp`结尾这样才能被`GetModelForName`正常解析。
2. 我得能自己指定fork服务器在哪儿启动。我加了一个`AFL_ENTRY_POINT`环境变量，在AFL的QEMU部分进行了解析。根据QEMU做重编译的过程，我们大概需要指明基
3. 在等fork的时候，加大超时时间的乘数。

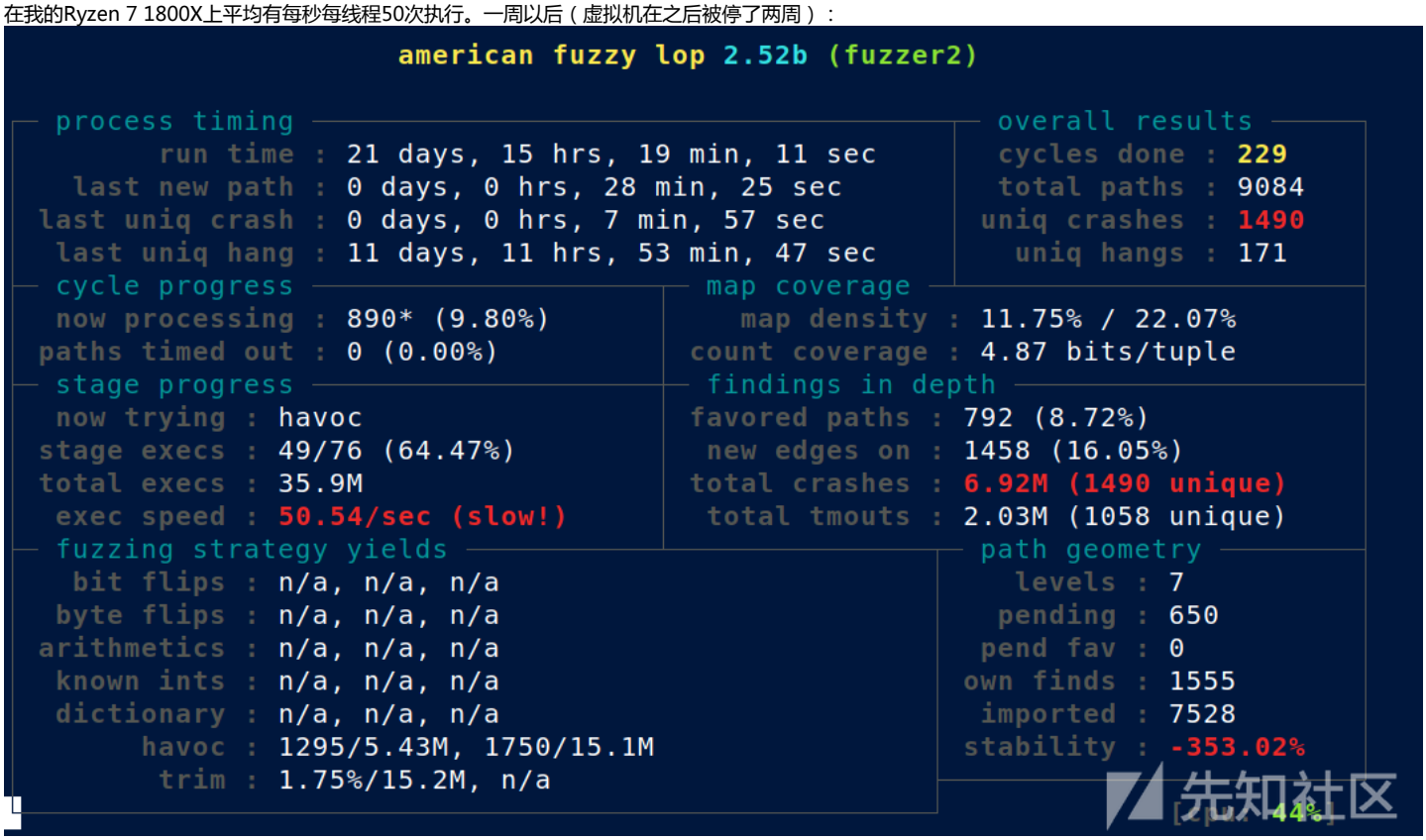
这些都patch了之后，跑fuzzer就很简单了：

```
$ export AFL_ENTRY_POINT=$(nm bspfuzz  |& grep forkserver | cut -d' ' -f1)
$ export AFL_INST_LIBS=1
$ afl-fuzz -m 2048 -Q -i fuzz/in -o fuzz/out -- ./bspfuzz @@
```

最好是用多进程，如果你直接用我的wrapper脚本的话，默认就可以。这是我用8核进行了5分钟fuzz之后的情况：



在我的Ryzen 7 1800X上平均有每秒每线程50次执行。一周以后（虚拟机在之后被停了两周）：



分流和造成原因分析

显然我们得找个办法把"好的"bug和没啥意思的bug分开（比如把纯粹的越界读）。我用了一个简单的基于调用栈的去重，然而在Valgrind里边跑了每一个样例。然后我grep `write`，非常的精妙。

```
$ sudo sysctl -w kernel.randomize_va_space=0
$ cd /path/to/bspfuzz/triage
$ ./triage.sh
$ ./valgrind.sh
```

```
$ egrep 'Invalid write' -A1 valgrind/* | egrep at | perl -n -e '/.*at (0x[^:]+)/ && print "$1\n";'
```

这得花点时间，我关了ASLR，所以这的crash位置都是唯一的。之后我又开了valgrind，然后手动把库的基地址都记下来，然后找到了每个"invalid write"的位置的库和偏移地址。

之后对于每个地址，我根据泄露的源码手动逆向了函数。有的地方是新的代码，但是前后的部分由于泄露的代码，极大的帮助了我逆向。我慢慢把大部分BSP解析代码都标上

对于每个poc，我验证了他们在windows的客户端上也可以触发。我发现所有bug在linux服务器和windows客户端上都有问题。

一点经验

从这个小项目中我个人学到的经验：

- AFL的QEMU模式对于攻击一小撮代码来说非常灵活，只要你稍微做点hacking，然后用个wrapper文件。
- 输入文件大小非常关键。从300k降到16k我得到了5倍的性能提升，如果再小点可能效果更好。
- 在整理从来没有fuzz过的代码的时候，分流就很重要了。
- 堆上的内存损坏不是安全问题 [滑稽]

示例bug：在CVirtualTerrain::LevelInit里的堆溢出

(这就是那个我发给Valve的报告。但是是个WONTFIX，也就是说，只要没人拿出exp，这个就一直是个0day)

在`CVirtualTerrain::LevelInit`里有个堆溢出，因为`dphysdisp_t::numDisplacements`变量的值可以比`g_DispCollTreeCount`大，在release版本里没有assert

```
void LevelInit( dphysdisp_t *pLump, int lumpSize )
{
	if ( !pLump )
	{
		m_pDispHullData = NULL;
		return;
	}
	int totalHullData = 0;
	m_dispHullOffset.SetCount(g_DispCollTreeCount);
	// [[ 1 ]]
	Assert(pLump->numDisplacements==g_DispCollTreeCount);
	// ■■lump■■■
	unsigned short *pDataSize = (unsigned short *)(pLump+1);
	for ( int i = 0; i < pLump->numDisplacements; i++ )
	{
		if ( pDataSize[i] == (unsigned short)-1 )
		{
			m_dispHullOffset[i] = -1;
			continue;
		}
		// [[ 2 ]]
		m_dispHullOffset[i] = totalHullData;
		totalHullData += pDataSize[i];
	}
```

在[[1]]位置的assert在release版本中没有，所以在[[2]]位置有一个溢出。需要注意的是`g_DispCollTreeCount`和`numDisplacements`值，也就是`pDataSize`的内容是一7里边很多模块都没有打开ASLR。

[我还附上了`numDisplacements = 0xffff`以及`g_DispCollTreeCount = 2`的BSP文件，可以有效的把`csgo.exe`搞崩掉。]

原文链接

https://phoenhex.re/2018-08-26/csgo-fuzzing-bsp

现在登录

热门节点

技术文章

社区小黑板

目录

RSS 关于社区 友情链接 社区小黑板