# 构建自己的FUZZ系统

J0hn | FreeBuf公开课最受白帽欢迎金牌讲师

# 个人介绍

- 常年研究自动化漏洞挖掘技术
- 深扎安全第一线

# FUZZ

FUZZ在漏洞挖掘中比较简单和粗暴,试错成本低,所以:”不管什么,先fuzz一把看看”

开源FUZZ列表

- VUzzer: 基于应用感知的自进化模糊工具
- AFL: 采用编译时检测和遗传算法的模糊工具
- Wadi-Fuzzer: 基于web浏览器语法的模糊器,是NodeFuzz的模块
- HonggFuzz: 基于代码覆盖率的反馈驱动的模糊工具
- CollAFL: 解决了afl中bitmap路径冲突问题,采用选择seed策略
- Peach:基于生成和基于突变的模糊测试

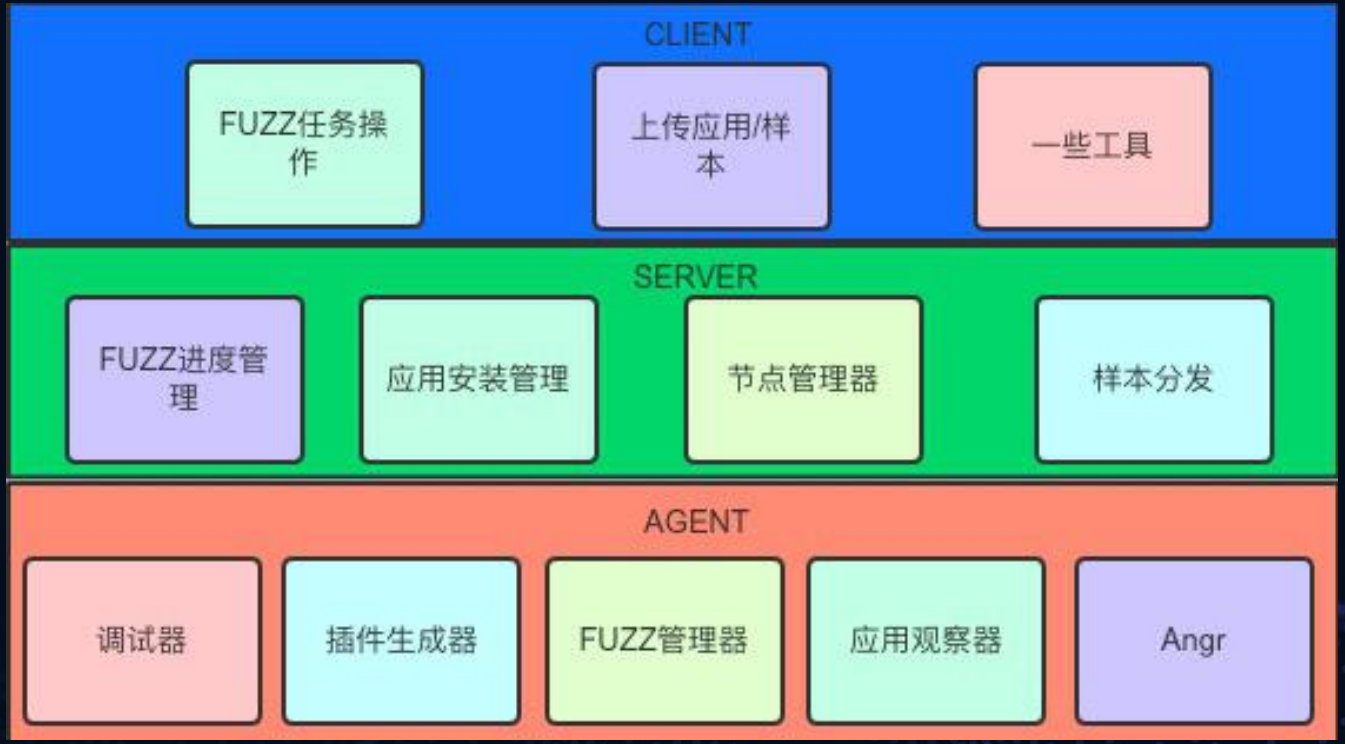
# FUZZ

使用FUZZ挖掘漏洞从运行-发现Carch-验证Carch的整个过程是long long a time

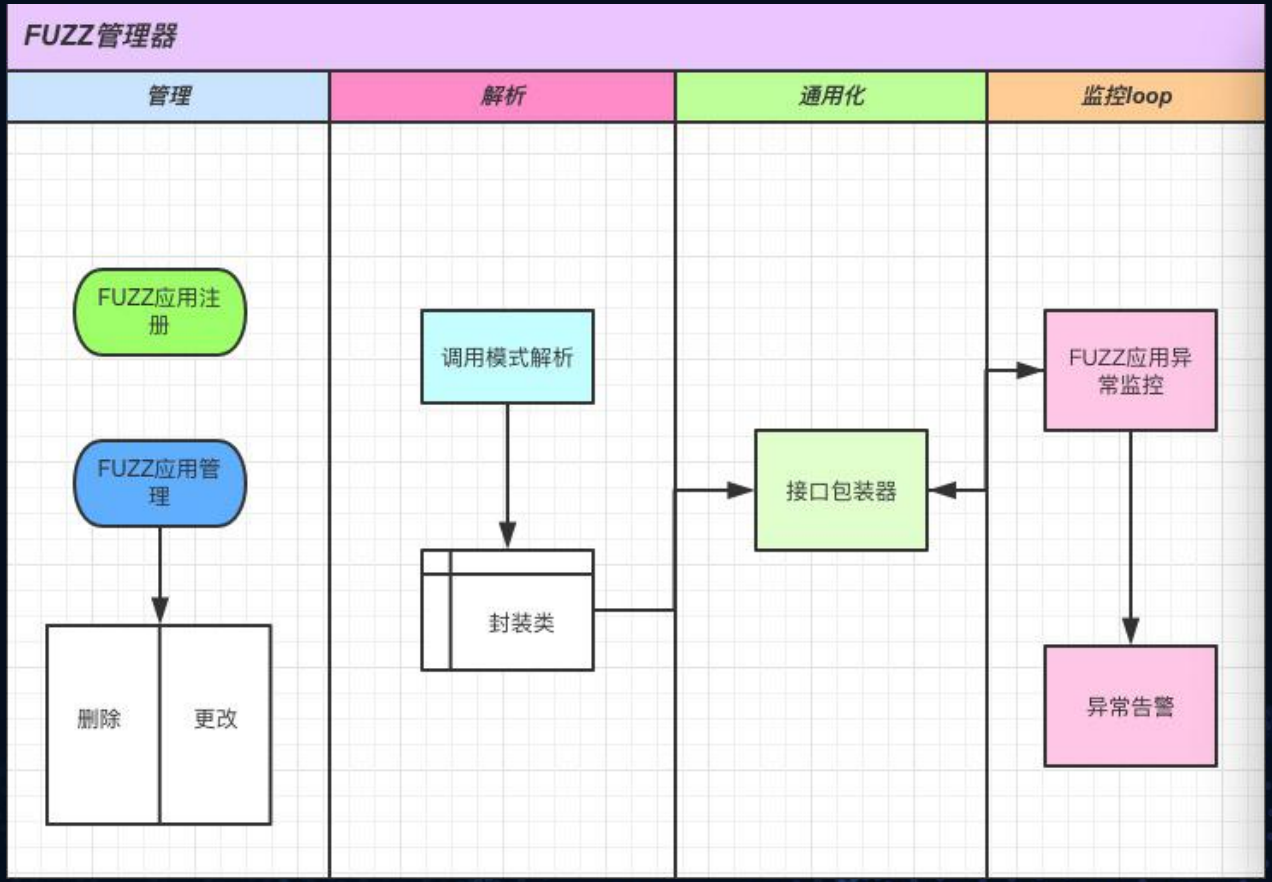
我们有什么办法能解决这些问题

# FUZZ基础架构

基于以上的思考



# 聊一聊Agent — FUZZ管理器



# 聊一聊Agent — FUZZ管理器

通过RPC注册

```
self._rpc_methods_ = ['importFuzz','boofuzzRun','boofuzzStat','zzufRun','zzufStat']
```

```
self._serv = SimpleXMLRPCServer((self.address_url, self.address_port), allow_none=True)
for name in self._rpc_methods_:
    self._serv.register_function(getattr(self, name))
```

调用模式

```
if fuzz_info['core'] == "boofuzz":
    def __setSession__(self):
        timestamp = datetime.datetime.utcnow().replace(microsecond=0).isoformat().replace(':', '-')

        connection = SocketConnection(self.host_, self.port_, proto=self.proto_, name=(timestamp,process_name))
        target = Target(connection=connection)
        self.session_ = Session(target=target, fuzz_name=timestamp)

    __setSession__()

    #self.FuzzStart(fuzz_info)

    self.session_.fuzz()
```

# 聊一聊Agent — FUZZ管理器

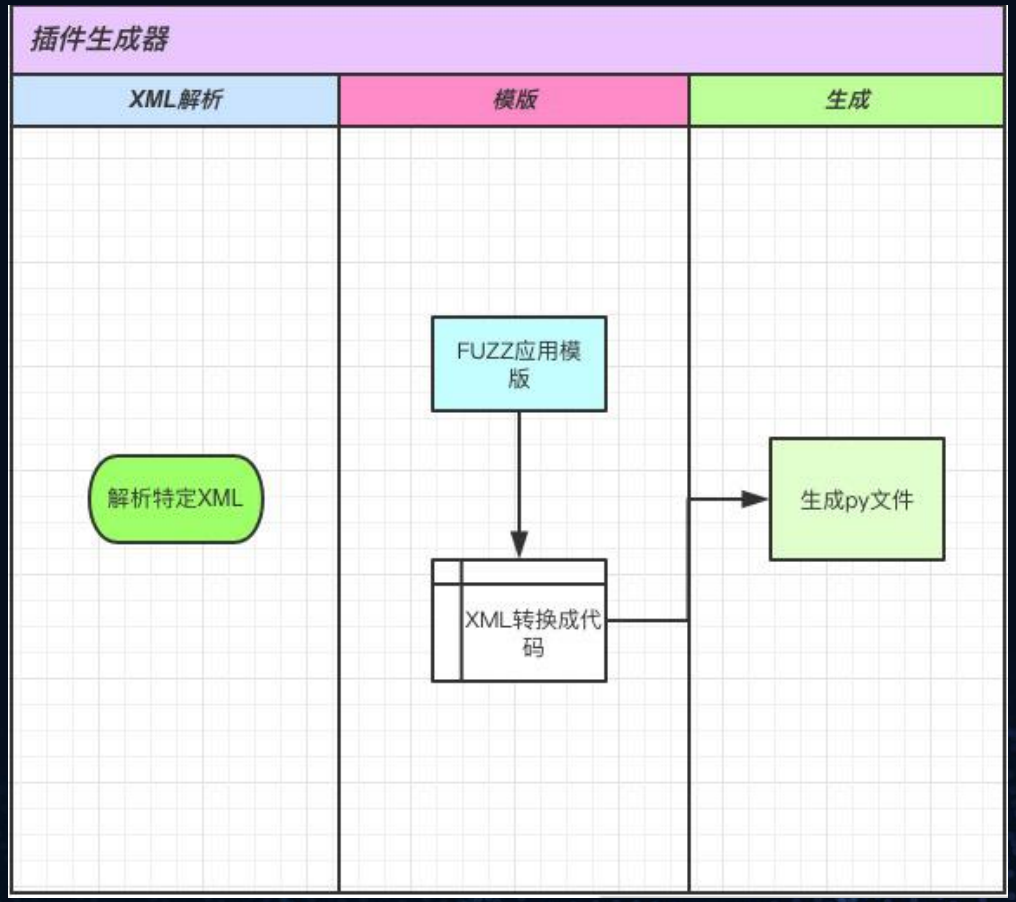
接口包装器

```
class FuzzBaseClass():
    def __init__(self):
        self.host_ = ""
        self.port_ = 0
        self.proto_ = ""
        self.session_ = None


    def FuzzStart(self,fuzz_info = {}):
        raise NotImplementedError
```

消息通知

```
def send_info(self,msg):
    self.server.getInfo(msg)

def send_log(self,process_name):
    if os.path.exists("/tmp/" + process_name + ".fuzz"):
        return None
    try:
        f = open("/tmp/" + process_name + ".fuzz", 'w')
        f.close()
    except:
        return False

    return True
```

# 聊一聊Agent — 插件生成器



# 聊一聊Agent — 插件生成器

```
<?xml version="1.0" encoding="iso-8859-1"?>
<fuzz tool_name="boofuzz">
    <initialize name="user">
        <info key="static" value="USER"></info>
        <info key="delim" value=" "></info>
        <info key="string" value="%%fuzz_info['user'].encode('ascii')%%"></info>
        <info key="static" value="\r\n"></info>
    </initialize>

    <initialize name="pass">
        <info key="static" value="PASS"></info>
        <info key="delim" value=" "></info>
        <info key="string" value="%%fuzz_info['pass'].encode('ascii')%%">></info>
        <info key="static" value="\r\n"></info>
    </initialize>

    <initialize name="stor">
        <info key="static" value="STOR"></info>
        <info key="delim" value=" "></info>
        <info key="string" value="AAAA"></info>
        <info key="static" value="\r\n"></info>
    </initialize>


    <session>
        <connect params="user"></connect>
        <connect params="pass,stor"></connect>

    </session>
</fuzz>
```

# 聊一聊Agent — 插件生成器

解析xml

```
def boofuzz(self):
    if self.xml_list[0][1] != "fuzz":
        return None
    if self.xml_list[0][2]['tool_name'] != "boofuzz":
        return None
    func = []
    for x in self.xml_list[1:]:
        node_name = x[1]
        node_value = x[2]
        if node_value == {}:continue;
        if node_name == 'initialize':
            node_name = '''s_initialize("%s")''' % node_value['name']
        elif node_name == 'info':
            if node_value['value'][0] == "%" and node_value['value'][-1] == "%":
                node_value['value'] = node_value['value'][2:-2]
                node_name = '''s_%s(%s)''' % (node_value['key'], node_value['value'])
            else:
                node_name = '''s_%s("%s")''' % (node_value['key'],node_value['value'])
        elif node_name == 'connect':
            params = ""
            if node_value['params'].find(",") > -1:
                tmp_val = node_value['params'].split(',')
                for val in tmp_val:
                    params += '''s_get("%s"),''' % val
                params = params[:-1]
            else:
                params = '''s_get(%s)''' % node_value['params']
            node_name = '''self.session_.connect(%s)''' % params

        func.append(node_name)
    return func
```

# 聊一聊Agent — 插件生成器



应用模版

```
boofTemp = '''from FrameWork import *
class Fuzz(FuzzBaseClass):
    def __init__(self):
        FuzzBaseClass.__init__(self)
    def FuzzStart(self,fuzz_info):
        %s
'''
```

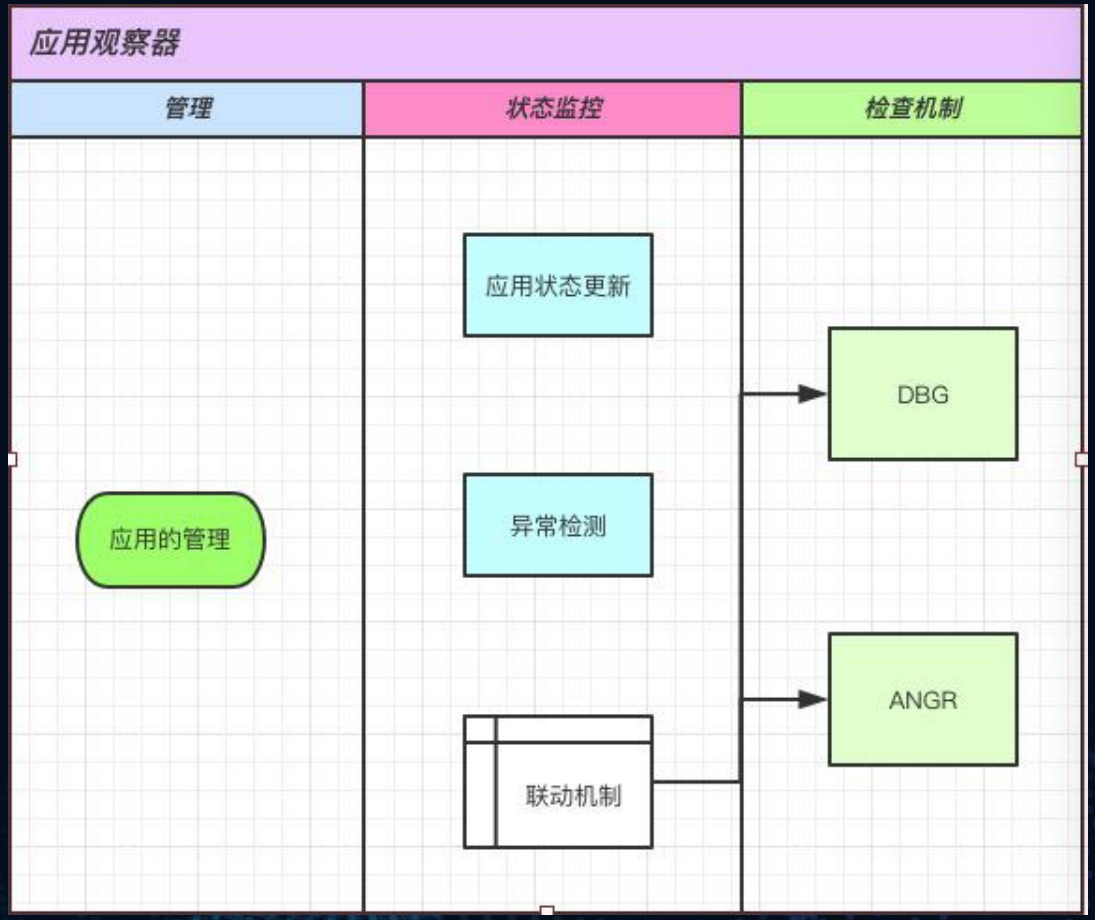
代码生成1

```
def str_to_tmpe(self,func):
    func_str = ""
    if len(func) == 0:
        return None
    for f in func:
        func_str += f + "\r\n            "

    return boofTemp % func_str
```

代码生成2

```
def importFuzz(self,plugin_name,load_dict):
    fdata = FuzzData()
    plugin_path = os.getcwd() + "/Plugin/" + plugin_name
    fdata.load_xml_file(plugin_path + ".xml")
    if load_dict['core'] == "fuzz":
        func = fdata.boofuzz()
    else:
        func = None

    x = fdata.str_to_tmpe(func)
    if os.path.exists(plugin_path + ".py"):
        os.remove(plugin_path + ".py")

    with open(plugin_path + ".py", "w") as f:
        f.write(x)

    plugin_name = 'Plugin.%s' % plugin_name
    self.import_str = importlib.import_module(plugin_name)
    self.load_dict = load_dict
```

# 聊一聊Agent — 应用观察器



# 聊一聊Agent — 应用观察器

## 应用启动

```
def boofuzzRun(self,process_name,cmd = ""):
    self.run_info[process_name] = {}
    self.run_info[process_name]['isrun'] = False
    if cmd == "":
        pid_ = monitor_process(process_name)
        if pid_ == None:
            return False

        self.run_info[process_name]['p'] = None
        self.run_info[process_name]['cmd'] = cmd
        self.run_info[process_name]['pid'] = pid_
        self.run_info[process_name]['isrun'] = True
        self.run_info[process_name]['runtime'] = 0
        self.run_info[process_name]['poll'] = None
    else:
        p = subprocess.Popen(cmd, stdin=subprocess.PIPE, stderr=subprocess.PIPE, close_fds=False,shell=True)
        self.run_info[process_name]['p'] = p
        self.run_info[process_name]['cmd'] = cmd
        self.run_info[process_name]['pid'] = int(p.pid)
        self.run_info[process_name]['isrun'] = True
        self.run_info[process_name]['runtime'] = time.time()
        self.run_info[process_name]['poll'] =p.poll()

    #print(self.run_info[process_name]['pid'])
    start = self.import_str.Fuzz()
    t = pro(target=start.Run, args=(self.load_dict,))
    #start.Run(self.load_dict)
    t.start()
```

# 聊一聊Agent — 应用观察器

应用状态跟踪

```
def boofuzzStat(self,process_name):
    process = self.run_info[process_name]
    #p = process['poll']
    pid = process['pid']
    cmd = process['cmd']
    p = Process_(pid)
    #pid_ = monitor_process(cmd)

    self.run_info[process_name]['poll'] = p.status()

    # if pid_ == None and int(pid_) != int(pid):
    #     return False

    comm = Comm()
    if comm.send_log(process_name) == True:
        os.remove("/tmp/" + process_name + ".fuzz")
        return None

    self.run_info[process_name]['exe'] = p.exe()
    self.run_info[process_name]['pwd'] = p.pwd()
    self.run_info[process_name]['uids'] = p.uids()
    self.run_info[process_name]['gids'] = p.gids()
    self.run_info[process_name]['connectios'] = p.connectios()
    self.run_info[process_name]['num_threads'] = p.num_threads()

    #self.run_info[process_name]['runtime'] = False

    return self.run_info[process_name]
```

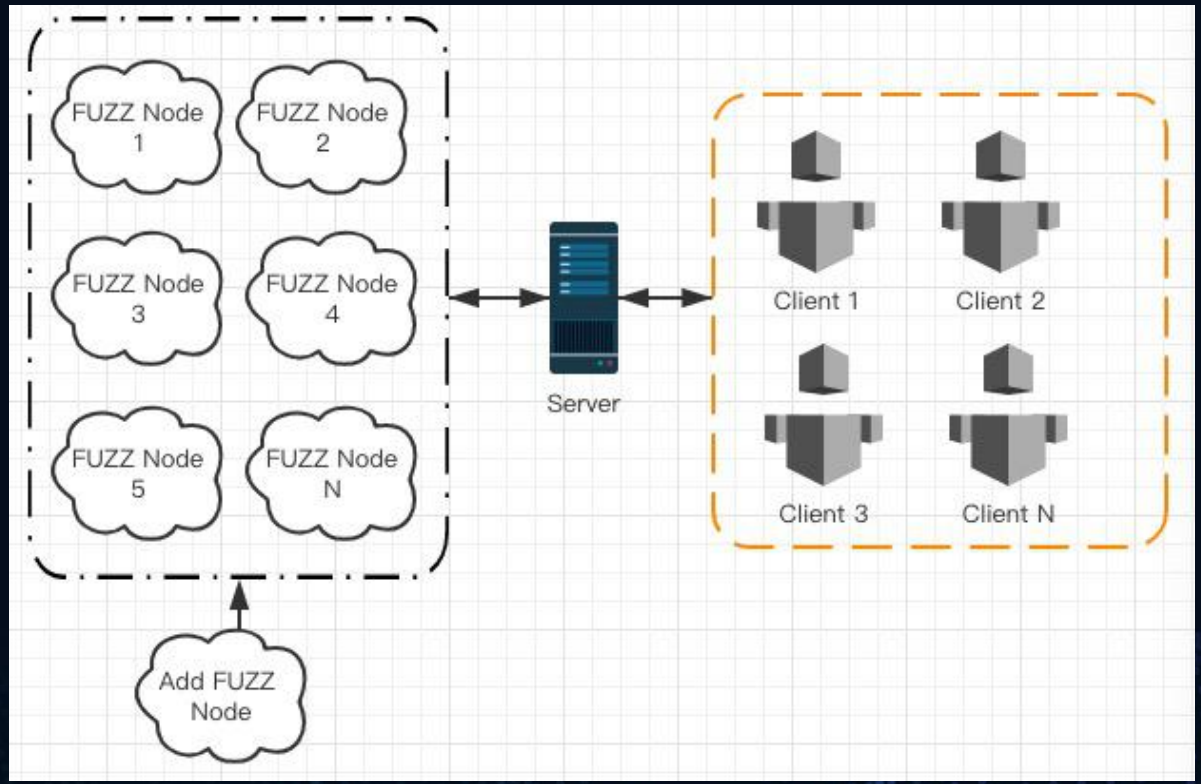
# 聊一聊Agent — 应用观察器

Angr相关

```
class Init:
    def __init__(self, project_name, args=None, main_opts={}, lib_opts={}):
        self.proj = angr.Project(project_name, main_opts=main_opts, lib_opts=lib_opts)
        project_info['arch'] = self.proj.arch
        project_info['entry'] = hex(self.proj.entry)
        project_info['filename'] = self.proj.filename
        project_info['bits'] = self.proj.arch.bits
        project_info['bytes'] = self.proj.arch.bytes
        project_info['shared_objects'] = self.proj.loader.shared_objects
        project_info['all_objects'] = self.proj.loader.all_objects
        project_info['main_objects'] = self.proj.loader.main_object
        project_info['min_addr'] = self.proj.loader.min_addr
        project_info['max_addr'] = self.proj.loader.max_addr
        project_info['execstack'] = self.proj.loader.main_object.extern_object
        project_info['pic'] = self.proj.loader.main_object.pic

    def blocks_search(self, entry):
        block = self.proj.factory(entry)
        return block.pp(), block.instructions, block.instructions_addrs

    def get_symbol(self):
        return self.proj.loader.main_object.imports

    def get_simstate(self):
        state = self.proj.factory.entry_state()
        return state
```

# 其他

所有工作都通过RPC进行通信




它是这样工作的

# Q & A

git地址: https://gitee.com/j0hn_shi/fconfuzz

FREEBUF | FIT

# THANKS