过滤型插件与反向代理的火花

2015年8月

Zero@BugScan

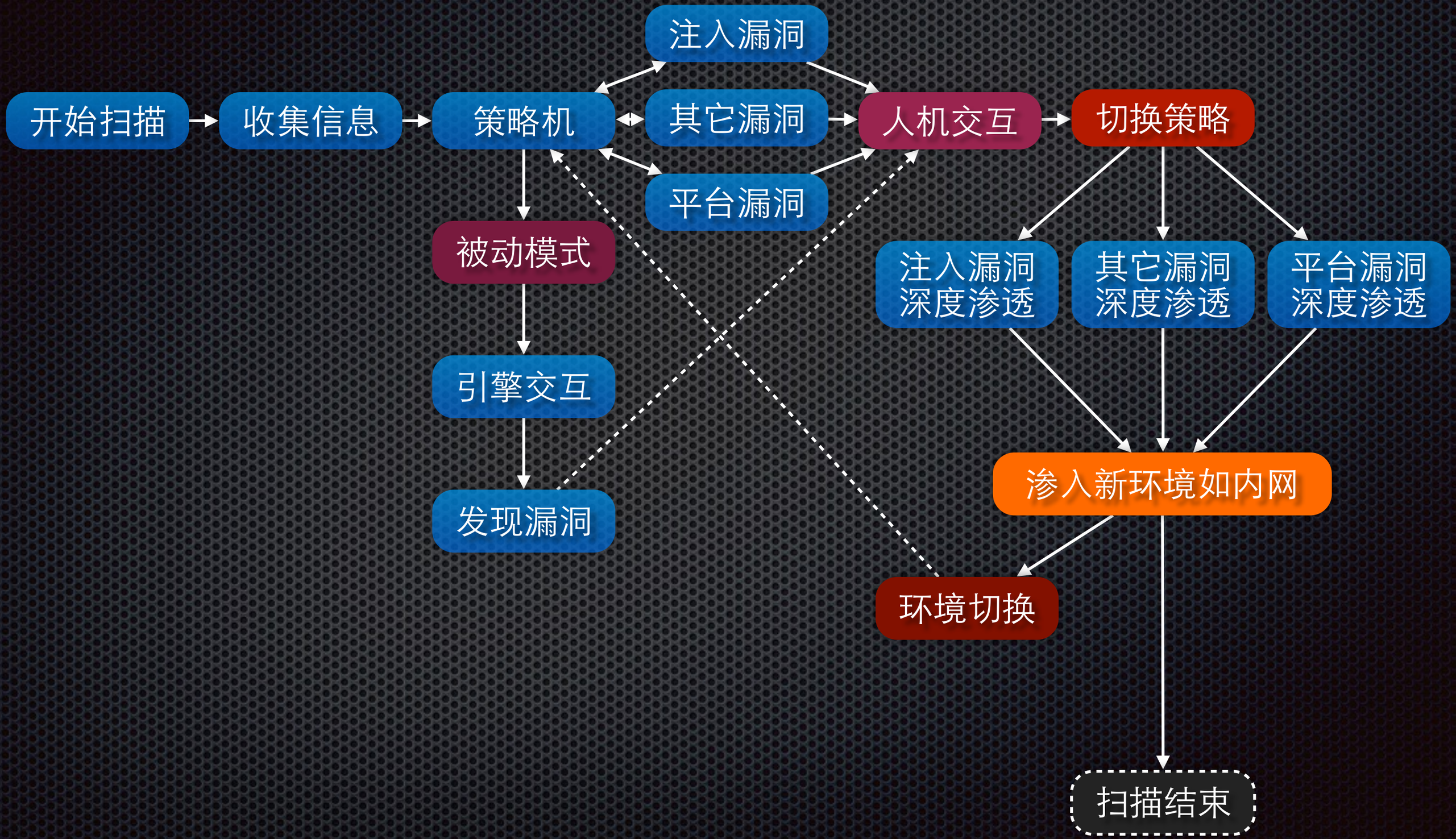
当前

- 国内外现有的软件模式：
  - 客户端模式，如WVS，Nessus
  - 在线(B/S)模式，如BugScan
  - 针对特定漏洞的扫描器，如Sqlmap，Metasploit
- 缺乏的模式和技术
  - 主动扫描与被动扫描结合模式
  - 扫描阶段的人机交互模式

BugScan可以实现的...

- 主被动结合
  - 主动扫描的同时，与被动扫描进行交互
- 扫描阶段的交互模式
  - 扫描流程控制中的人工决策
  - 人工渗透过程中的引擎支持

理想架构


注入漏洞
开始扫描
收集信息
策略机
其它漏洞
人机交互
切换策略
平台漏洞
被动模式
注入漏洞
深度渗透
其它漏洞
深度渗透
平台漏洞
深度渗透
引擎交互
渗入新环境如内网
发现漏洞
环境切换
扫描结束

一个插件，一个桥梁…

```python
def assign(service, arg):
    if service == 'www' and (arg.endswith('?bproxy') or arg.endswith('?bproxy/')):
        return True, None, 'proxy'

def audit(arg):
    try:
        HTTPProxyServer('0.0.0.0', 8080)
        while _run:
            asyncore.loop(count=1)
    except:
        pass
    security_note('Stop success, Process %d links.' % _count)

if __name__ == '__main__':
    from dummy import *
    audit(assign('www', 'www.baidu.com/?bproxy')[1])
~
```

添加任务时的目标以?bproxy 或者?bproxy/结尾则启动代理

```python
        if self.url.startswith('http://') and util.get_url_ext(self.url).lower() not in FORBIDDEN_MIME_TYPE:
            debug('[>>>] %s', self.url)
            if self.url.endswith('?bstop'):
                debug('[***] Close HTTP Proxy')
                _run = False
                return
            host = util.get_url_host(self.url)
            if host == util._G['target'] or (util._G['subdomain'] and is_subdomain_of(host, util._G['target'])):
                task_push('www', self.url)

```

手动停止

交给Bugscan进行安全审计

应用场景

- 手机或PC APP的安全审计(手机设置代理)
- 网站(爬虫)不可见应用的检测(Flash, 内部系统)

可扩展的联想…

- 可以做成chrome插件与BugScan交互(插件跑RPCServer)
- 插件完全可以通过运行HTTP服务生成一个交互性的UI
- BurpSuite的大部分功能，可以在BugScan里实现
- 与PhantomJS配合征服web 2.0(其它HeadLess浏览器也可以)
- 总之…BugScan的框架提供了一个万能钥匙

扫描器的未来形态

- 全能的系统
- 可以存在任何环境
- 可以做为一个强大的后端引擎
- 可以提供丰富的前端工具
- 是扫描器的最终形态(扫描器界的eclipse)

心里话…

- 学习安全是在学习一种思维模式
- 是一种可以受益终身，跨行业的模式