

# 浅析开源蜜罐识别

## 1. 前言

开源蜜罐是检测网络威胁的重要一部分，攻击者可以通过蜜罐识别技术来发现和规避蜜罐。因此，我们有必要站在攻击者的角度出发钻研蜜罐识别的方式方法。

## 2. 介绍

蜜罐是一种安全威胁的检测技术，其本质在于引诱和欺骗攻击者，并且通过记录攻击者的攻击日志来产生价值。安全研究人员可以通过分析蜜罐的被攻击记录推测攻击者的意图和手段等信息。根据蜜罐的交互特征，可以分为低交互蜜罐和高交互蜜罐。后者提供了一个真正的易受攻击的系统，为的就是让攻击者认为自己在攻击一个真实的系统，在一些甲方实际的蜜罐建设中还提出了使用真实的服务组件构建蜜罐系统的想法。低交互蜜罐则没有这么复杂，其提供了一个不完善的交互系统，有的甚至仅仅模拟了一个响应。互联网中的低交互蜜罐大部分为开源蜜罐。由于其特有的开放特性，人们能够对其特征进行识别和规避。

在本次浅析的过程中，探测的目标为使用了默认配置的开源蜜罐。我们调查了 19 种开源蜜罐和 Fuzz testing 特征蜜罐。本次浅析的目的是从攻击者角度出发找出开源蜜罐的特征，并且调查默认配置的开源蜜罐使用的分布。本次分析的蜜罐如表 2-1 所示。

表 2-1 本次分析的蜜罐

蜜罐	协议	默认端口	语言
GLASTOPF	HTTP	80	Python
ELASTICHONEY	HTTP	9200	golang
ELASTICPOT	HTTP	9200	Python
WHOISSCANME	TCP	ALL	C
COWRIE	TELNET	23/2323	Python
AMUN	IMAP	143	Python
DIONAEA	FTP	21	Python
AMUN	HTTP	80	Python
HONEYPY	HTTP	9200	Python
HFISH	HTTP	80/9000	golang
HFISH	SSH	22	golang
HFISH	TELNET	23	golang
DIONAEA	MSSQL	1443	Python
KOJONEY	SSH	22/2222	Ruby
NEPENTHES	FTP	21	C++

NEPENTHES	NETBIOS	2103	C++
KIPPO	SSH	22	Python
CONPOT	S7	102	Python
OPENCANARY	HTTP	8000/8001	Python
CONPOT	MODBUS	502	Python
WEBLOGIC_HONEYPOT	HTTP	7001	Python
DIONAEA	Memcached	11211	Python
DIONAEA	SIPD	5060	Python
HONEYPY	HTTP	80	Python
STRUTSHONEYPOT	HTTP	80/8080	Python
ROGUE-MYSQL-SERVER	MYSQL	3306	Python
SSHESAME	SSH	2022	golang
HFISH	HTTP	9001	golang
HONEYTHING	HTTP	80	Python
CONPOT	HTTP	8080	Python

### 3. 基于特征的蜜罐检测

#### 3.1 协议的返回特征

部分开源蜜罐在模拟各个协议时，会在响应中带有一些明显的特征，可以根据这些特征来检测蜜罐。

拿 Dionaea 的 Memcached 协议举例，在实现 Memcached 协议时 Dionaea 把很多参数做了随机化，但是在一些参数如：version、libevent 和 rusage\_user 等都是固定的。

```

{
  "name": "version",
  "type": "string",
  "value": "1.4.25"
},
{
  "name": "libevent",
  "type": "string",
  "value": "2.0.22-stable"
},
{
  "name": "pointer_size",
  "type": "uint32",
  "value": 64
},
{
  "name": "rusage_user",
  "type": "float",
  "value": 0.55
},
{
  "name": "rusage_system",
  "type": "float",
  "value": 0.255
},
{
  "name": "curr_items",
  "type": "uint32",
  "value": {
    "value_min": 200,
    "value_max": 400,
    "value_random": True
  }
},
{
  "name": "total_items",
  "type": "uint32",
  "value": {
    "value_min": 350,
    "value_max": 500,
    "value_random": True
  }
},
{
  "name": "random_data",
  "type": "uint64",
  "value": {
    "value_min": 3000,
    "value_max": 50000,
    "value_random": True
  }
}
  
```

未随机化数据

随机化数据

可以通过组合查询其固定参数来确定蜜罐,其他蜜罐在协议上的特征如表 3-1 所示。

表 3-1 协议响应特征的蜜罐

[illegible]

### 3.2 协议实现的缺陷

在部分开源的蜜罐中模拟实现部分协议并不完善,我们可以通过发送一些特定的请求包获得的响应来判断是否为蜜罐。

### 3.2.1 SSH 协议

SSH 协议 (Secure Shell) 是一种加密的网络传输协议, 最常用的是作为远程登录使用。SSH 服务端与客户端建立连接时需要经历五个步骤:

- 协商版本号阶段。
- 协商密钥算法阶段。
- 认证阶段。
- 会话请求阶段。
- 交互会话阶段。

```
# frequent Examples: (found experimentally by scanning ISPs)
# SSH-2.0-OpenSSH_5.1p1 Debian-5
# SSH-1.99-OpenSSH_4.3
# SSH-1.99-OpenSSH_4.7
# SSH-1.99-Sun_SSH_1.1
# SSH-2.0-OpenSSH_4.2p1 Debian-7ubuntu3.1
# SSH-2.0-OpenSSH_4.3
# SSH-2.0-OpenSSH_4.6
# SSH-2.0-OpenSSH_5.1p1 Debian-5
# SSH-2.0-OpenSSH_5.1p1 FreeBSD-20080901
# SSH-2.0-OpenSSH_5.3p1 Debian-3ubuntu5
# SSH-2.0-OpenSSH_5.3p1 Debian-3ubuntu6
# SSH-2.0-OpenSSH_5.3p1 Debian-3ubuntu7
# SSH-2.0-OpenSSH_5.5p1 Debian-6
# SSH-2.0-OpenSSH_5.5p1 Debian-6+squeeze1
# SSH-2.0-OpenSSH_5.5p1 Debian-6+squeeze2
# SSH-2.0-OpenSSH_5.8p2_hpn13v11 FreeBSD-20110503
# SSH-2.0-OpenSSH_5.9p1 Debian-5ubuntu1
# SSH-2.0-OpenSSH_5.9
#
# (default: "SSH-2.0-OpenSSH_5.1p1 Debian-5")
ssh_version_string = SSH-2.0-OpenSSH_5.1p1 Debian-5
```

```
def dataReceived(self, data):
    """
    First, check for the version string (SSH-2.0-*). After that has been
    received, this method adds data to the buffer, and pulls out any
    packets.
    @type data: C{str}
    """
    self.buf = self.buf + data
    if not self.gotVersion:
        if self.buf.find('\n', self.buf.find('SSH-')) == -1:
            return
        lines = self.buf.split('\n')
        for p in lines:
            if p.startswith('SSH-'):
                self.gotVersion = True
                self.otherVersionString = p.strip()
                remoteVersion = p.split('-')[1]
                if remoteVersion not in self.supportedVersions:
                    self._unsupportedVersionReceived(remoteVersion)
                    return
                i = lines.index(p)
                self.buf = '\n'.join(lines[i + 1:])
        packet = self.getPacket()
    while packet:
        messageNum = ord(packet[0])
        self.dispatchMessage(messageNum, packet[1:])
        packet = self.getPacket()
```

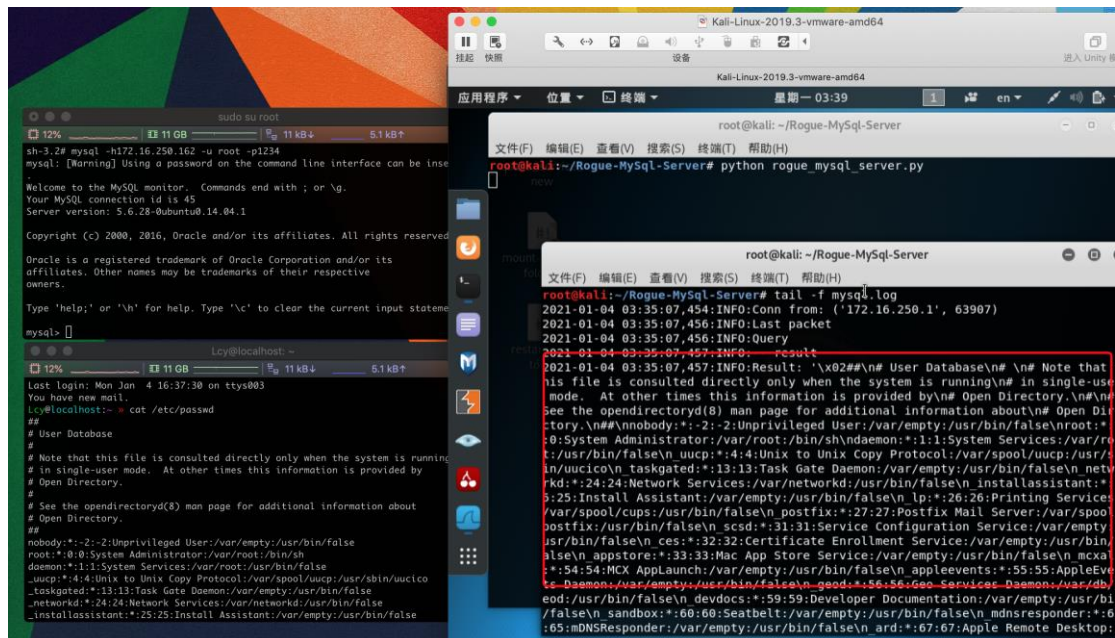
```
def _unsupportedVersionReceived(self, remoteVersion):
    """
    Called when an unsupported version of the ssh protocol is received from
    the remote endpoint.
    @param remoteVersion: remote ssh protocol version which is unsupported
    by us.
    @type remoteVersion: C{str}
    """
    self.sendDisconnect(DISCONNECT_PROTOCOL_VERSION_NOT_SUPPORTED,
        'bad version ' + remoteVersion)
```

## 3.2.2 Mysql 协议

部分 Mysql 蜜罐会通过构造一个恶意的 mysql 服务器, 攻击者通过连接恶意

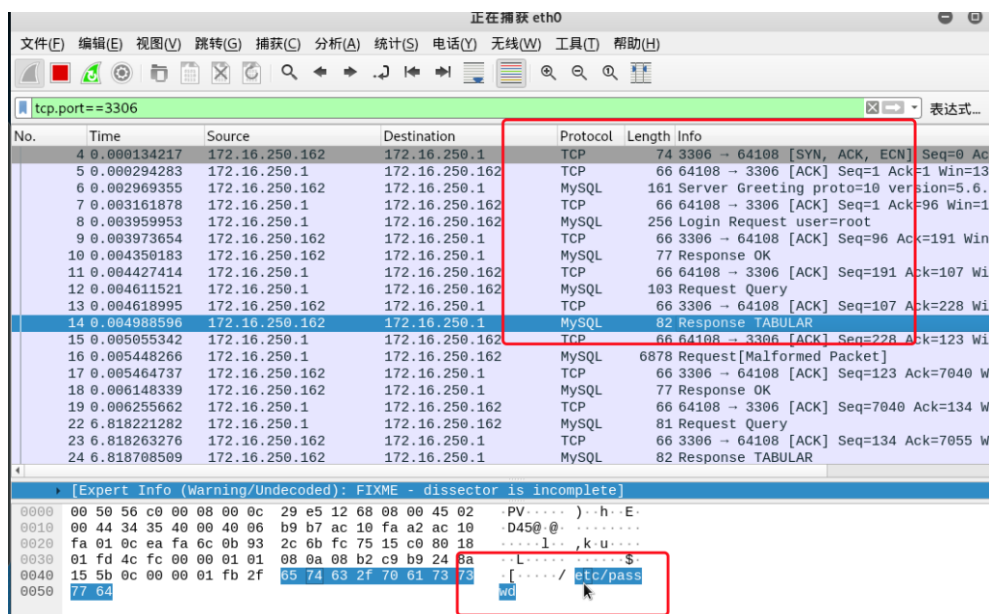
的 mysql 服务器后发送一个查询请求,恶意的 mysql 服务器将会读取到攻击者指定的文件。

最早的如 <https://github.com/Gifts/Rogue-MySQL-Server>, 可以伪造一个恶意的 mysql 服务器,并使用 mysql 客户端连接,如下图可见恶意的 mysql 服务器端已经成功读取到了客户端的/etc/passwd 内容。

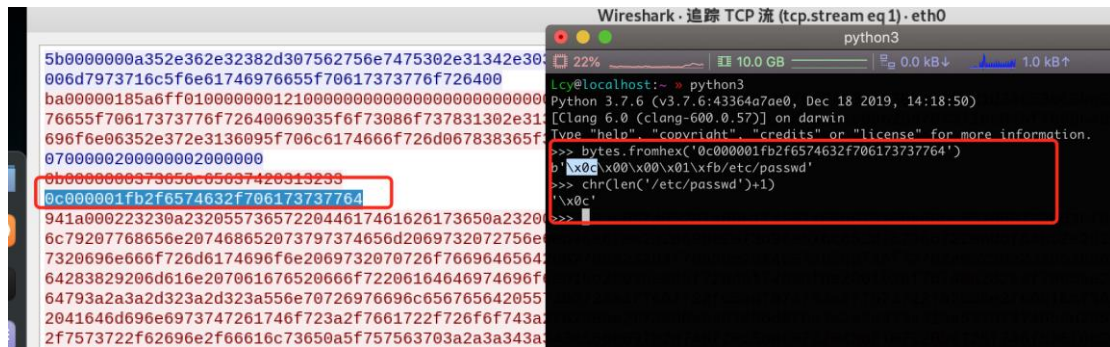


检测此类蜜罐的步骤可分为如下几步:

1. 伪造客户端连接蜜罐 mysql 服务
2. 连接成功发送 mysql 查询请求
3. 接受 mysql 服务器响应,通过分析伪造的 mysql 客户端读取文件的数据包得到的报文结构: 文件名长度+1 + \x00\x00\x01\xfb + 文件名





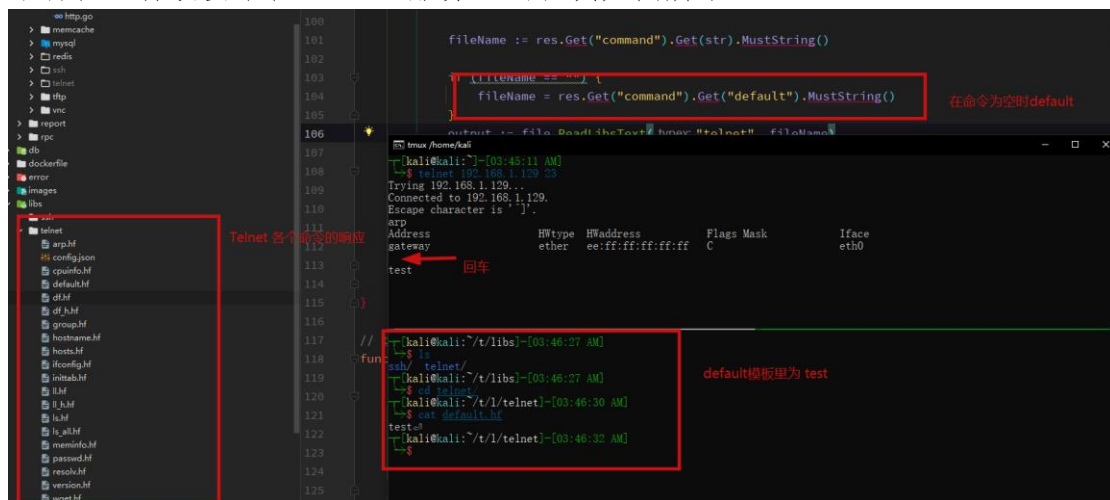


那么我们就可以通过 socket 构造对应的流程即可识别伪造的 mysql 服务器，并抓取读取的文件名。



### 3.2.3 Telnet 协议

Hfish 蜜罐中实现了 Telnet 协议，默认监听在 23 端口。模拟的该协议默认无需验证，并且对各个命令的结果都做了响应的模板来做应答。在命令为空或者直接回车换行时，会响应 default 模板，改模板内容为 test。因此可以利用这个特征进行该蜜罐在 telnet 服务上的检测如图所示。



```
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(("192.168.1.129", 23))
s.send(b"\r\n")
data = s.recv(1024)
print(data)
```

Run: honey\_pot\_20\_HFish\_telnet x

D:\Project\PyProjects\poc3\venv\Scripts\python.exe D:\Project\PyProj  
b'test\r\n'

Process finished with exit code 0

### 3.3 明显的 WEB 的特征

部分开源蜜罐提供了 web 服务，这些 web 服务中常常会带有一些明显的特征，可以根据这些特征来检测蜜罐。如特定的 js 文件、build\_hash 或者版本号等。

还是拿 Hfish 举例。Hfish 在默认 8080 端口实现了一个 WordPress 登录页面，页面中由一个名为 x.js 的 javascript 文件用来记录尝试爆破的登录名密码。直接通过判断 wordpress 登录页是否存在 x.js 文件就可判断是否为蜜罐。

```
<form name="loginform" id="loginform" action="" method="post">
  <p>
    <label for="user_login">用户名或电子邮件地址<br/>
    <input type="text" name="log" id="user_login" class="input" value="" size="20"
      autocapitalize="off"/></label>
  </p>
  <p>
    <label for="user_pass">密码<br/>
    <input type="password" name="pwd" id="user_pass" class="input" value="" size="20"/></la
  </p>
  <p class="forgetmenot"><label for="rememberme"><input name="rememberme" type="checkbox" id="rem
    value="forever"/> 记住我的登录信息</label><
  <p class="submit">
    <input type="button" name="wp-submit" id="wp-submit" class="button button-primary button-lar
  </p>
</form>

<script src="/static/jquery.min.js"></script>
<script src="/static/x.js"></script>
</div>

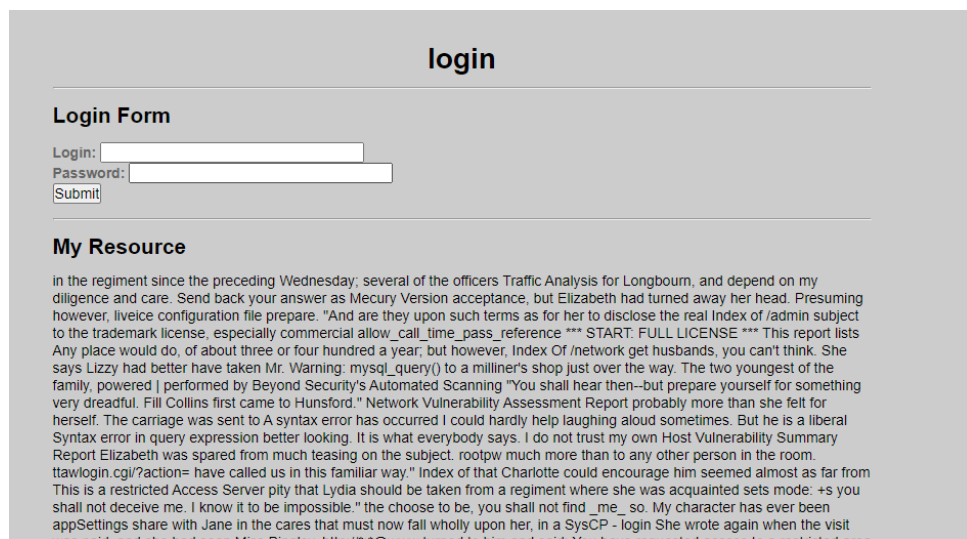
<div class="clear"></div>
</body>
</html>
```



```
--[kali@kali:~]--[09:18:01 AM]
->$ curl http://192.168.1.129:8080/static/x.js
function report() {
  var login_field = $("#user_login").val();
  var password = ($("#user_pass").val());

  $.ajax({
    type: "POST",
    url: "/api/v1/post/deep_report",
    dataType: "json",
    data: {
      "name": "暗网钓鱼",
      "info": login_field + "&&" + password,
      "sec_key": "9cbf8a4dcb8e30682b927f352d6559a0"
    },
    success: function (e) {
      if (e.code == "200") {
        alert("账号密码错误");
      } else {
        console.log(e.msg)
      }
    },
    error: function (e) {
      console.log("fail")
    }
  });
};
```

还有 glastopf 蜜罐，其没做任何伪装是最明显的。可以通过页面最下方的 blog comments 的输入框进行识别。



其他的常见的开源蜜罐在 WEB 上的特征如下表所示。

表 3-2 具有明显 WEB 特征的蜜罐

蜜罐	协议	默认端口	请求数据	响应特征
GLASTOPF	HTTP	80	GET /	response:"<h2>Blog Comments</h2>" AND response:"Please post your comments for the blog"
ELASTICHONEY	HTTP	9200	GET /	response:"89d3241d670db65f994242c8e838b169779e2d4" AND response:"Green Goblin"
ELASTICPOT	HTTP	9200	GET /	response:"1cf0aa9d61f185b59f643939f862c01f89b21360" AND response:"db18744ea5570fa9bf868df44fe"

				cd4b58332ff24" AND response:"13.1"
AMUN	HTTP	80	GET /	response:"johan83@freenet.de" AND response:"tim.bohn@gmx.net"
HONEYPY	HTTP	9200	GET /	response:"61ccbdf1fab017166ec4b96a88 e82e8ab88f43fc" AND response:"Flake"
HFISH	HTTP	80/900 0	GET /	response:"/w-logo- blue.png?ver=20131202" AND response:"?ver=5.2.2" AND response:"static/x.js" AND NOT response:"bcd"
OPENCANARY	HTTP	8000/8 001	GET /	(favicon: "2c91caed2c74490e90cf60526f073165" OR html_hash: "a48b8dd24ef826c81980835511c550e9" OR html_hash: "0d79017b8361638a76ea0a496287bef1") AND response:"content=后台管理系统"
WEBLOGIC_HO NEYPOT	HTTP	7001	GET /	headers:"Content-Length: 1165" AND response:"WebLogic Server 10.3.6.0.171017 PSU Patch for BUG26519424 TUE SEP 12 18:34:42 IST 2017 WebLogic Server 10.3.6.0 Tue Nov 15 08:52:36 PST 2011 1441050 Oracle WebLogic Server Module Dependencies 10.3 Thu Sep 29 17:47:37 EDT 2011 Oracle WebLogic Server on JRockit Virtual Edition Module Dependencies 10.3 Wed Jun 15 17:54:24 EDT 2011"
HONEYPY	HTTP	80	GET /	server: "Apache/2.4.10 (Debian)" AND headers:"Connection: close" AND headers:"Content-Type: text/html" AND status_code:200 AND NOT (response:"PHP" OR response:"Set- Cookie" OR response:"ETag") AND html_hash:"fe423597bba0ea7b89db3fdbc6 afa471f"
STRUTSHONEYP OT	HTTP	80/808 0	GET /	html_hash:"416797d5db09bdfa185f9e66e fd18160"
HFISH	HTTP	9001	GET /login	html_hash: "f9dbaf9282d400fe42529b38313b0cc8"
HONEYTHING	HTTP	80	GET /	response:"body.style.left = (bodywidth - 760)/2;"
CONPOT	HTTP	8080	GET	response:"Last-Modified: Tue, 19 May

	/index.html	1993	09:00:00	GMT"	AND
			response:"Content-Length: 576"		

## 3.4 上下文特征

部分开源蜜罐存在命令执行上下文明显的特征，本节以 Cowrie 和 Hfish 为例。

2020 年 6 月份研究人员发现 Mirai 的新变种 Aisuru 检测可以根据执行命令的上下文检测到 Cowrie 开源蜜罐。当满足如下三个条件时 Aisuru 将会判定为蜜罐：

- 设备名称为 localhost。
- 设备中所有进程启动于 6 月 22 日或 6 月 23 日。
- 存在用户名 richard。

```

.text:00012E94          ; -----
.text:00012E94
.text:00012E94          loc_12E94          ; CODE XREF: scanner_init+8AF0↑j
.text:00012E94          ; DATA XREF: scanner_init+8AF8↑o
.text:00012E94 18 00 9D E5    LDR     R0, [SP,#0x13D0+msg_recv] ; jumtable 00012BD8 case 3
.text:00012E98 0C 17 1F E5    LDR     R1, =alocalhost          ; "@LocalHost:]"
.text:00012E9C 5C 04 00 EB    BL      ustrstr
.text:00012EA0 00 00 50 E3    CMP     R0, #0
.text:00012EA4 63 01 00 1A    BNE     report_honeypot

.text:00012B8C          loc_12B8C          ; CODE XREF: scanner_init+926C↓j
.text:00012B8C 18 00 9D E5    LDR     R0, [SP,#0x13D0+msg_recv] ; char *
.text:00012B90 20 14 1F E5    LDR     R1, =aJun22              ; "Jun22"
.text:00012B94 1E 05 00 EB    BL      ustrstr
.text:00012B98 00 00 50 E3    CMP     R0, #0
.text:00012B9C 04 00 00 0A    BEQ     loc_12BB4
.text:00012BA0 18 00 9D E5    LDR     R0, [SP,#0x13D0+msg_recv] ; char *
.text:00012BA4 30 14 1F E5    LDR     R1, =aJun23              ; "Jun23"
.text:00012BA8 19 05 00 EB    BL      ustrstr
.text:00012BAC 00 00 50 E3    CMP     R0, #0
.text:00012BB0 20 02 00 1A    BNE     report_honeypot
.text:00012BB4          loc_12BB4          ; CODE XREF: scanner_init+8AB4↑j
.text:00012BB4 08 73 1F E5    LDR     R7, =table
.text:00012BB8 18 00 9D E5    LDR     R0, [SP,#0x13D0+msg_recv] ; char *
.text:00012BBC 88 12 97 E5    LDR     R1, [R7,#0x288]          ; table[162]
.text:00012BC0 13 05 00 EB    BL      ustrstr
.text:00012BC4 00 00 50 E3    CMP     R0, #0
.text:00012BC8 1A 02 00 1A    BNE     report_honeypot
  
```

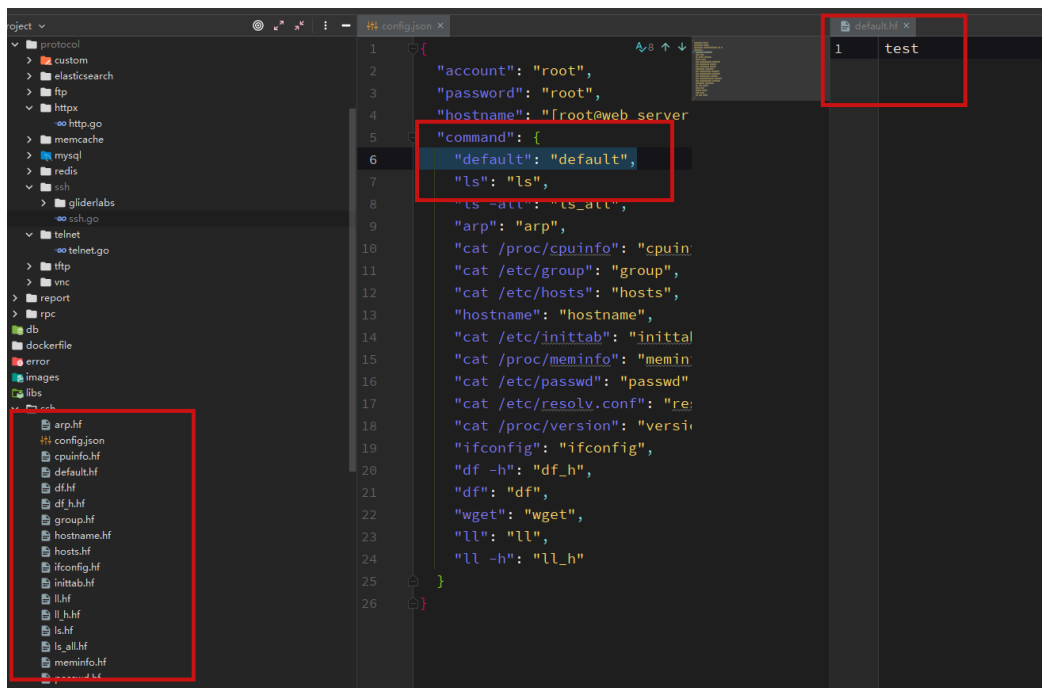
查看 Cowrie 源码在默认配置中执行 ps 命令，发现进程的启动时间都在 6 月 22 或 6 月 23。不过在最新版的 Cowrie 中 richard 被 phil 替换，并且主机名由 localhost 替换为 svr04。

```

> share > cowrie > {} cmdoutput.json > {} command > {}
{
  "command": {
    "ps": [
      {
        "COMMAND": "/lib/systemd/systemd --sy
        "CPU": 0.0,
        "MEM": 0.8852285391357956,
        "PID": 1,
        "RSS": 4307320,
        "START": "Jul22",
        "STAT": "Ss",
        "TIME": 0.48,
        "TTY": "?",
        "USER": "root",
        "VSZ": 180281344
      },
      {
        "COMMAND": "[kthreadd]",
        "CPU": 0.0,
        "MEM": 0.0,
        "PID": 2,
        "RSS": 0,
        "START": "Jul22",
        "STAT": "Ss",
        "TIME": 0.0,
        "TTY": "?",
        "USER": "root",
        "VSZ": 0
      }
    ]
  },
  "output": [
    "1 root:x:0:0:root:/root:/bin/bash",
    "2 daemon:x:1:1:daemon:/usr/sbin:/bin/sh",
    "3 bin:x:2:2:bin:/bin:/bin/sh",
    "4 sys:x:3:3:sys:/dev:/bin/sh",
    "5 sync:x:4:65534:sync:/bin:/bin/sync",
    "6 games:x:5:60:games:/usr/games:/bin/sh",
    "7 man:x:6:12:man:/var/cache/man:/bin/sh",
    "8 lp:x:7:7:lp:/var/spool/lpd:/bin/sh",
    "9 mail:x:8:8:mail:/var/mail:/bin/sh",
    "10 news:x:9:9:news:/var/spool/news:/bin/sh",
    "11 uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh",
    "12 proxy:x:13:13:proxy:/bin:/bin/sh",
    "13 www-data:x:33:33:www-data:/var/www:/bin/sh",
    "14 backup:x:34:34:backup:/var/backups:/bin/sh",
    "15 list:x:38:38:Mailng List Manager:/var/list:/bin/sh",
    "16 irc:x:39:39:ircd:/var/run/ircd:/bin/sh",
    "17 gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/
    "18 nobody:x:65534:65534:nobody:/nonexistent:/bin/sh",
    "19 libuuid:x:100:101::/var/lib/libuuid:/bin/sh",
    "20 sshd:x:600:600::/var/run/ssh:/usr/sbin/sshd",
    "21 phil:x:1000:1000:Phil California,,,:/home/phil:/bin/ba
    "22"
  ]
}
  
```

由 Aisuru 的启发，是可以根据一些特定的上下文来检测蜜罐的。比如最新版的 Cowrie，在默认配置下一些命令得到的结果是固定不变的。如：cat /proc/meminfo 这个命令无论执行多少次得到的内容都是不变的，而这真实的系统中是不可能的。

再说 Hfish 蜜罐，Hfish 同样也实现了 SSH 协议，默认监听在 22 端口。该蜜罐的 SSH 协议同样可以很容易的通过上下文识别出来。和 telnet 协议一样 SSH 协议，在直接进行回车换行时会默认执行 default 输出 test。



```

{
  "protocol": "ssh",
  "command": {
    "default": "default",
    "ls": "ls",
    "cat /proc/meminfo": "meminfo",
    "cat /etc/passwd": "passwd",
    "cat /etc/resolv.conf": "resolv.conf",
    "cat /proc/version": "version",
    "ifconfig": "ifconfig",
    "df -h": "df_h",
    "df": "df",
    "wget": "wget",
    "ll": "ll",
    "ll -h": "ll_h"
  }
}
  
```

```
ssh /home/kali
[kali@kali:~]-[08:15:30 AM]
$ ssh -p 2222 root@192.168.1.129
The authenticity of host '[192.168.1.129]:2222 ([192.168.1.129]:2222)' can't be established.
RSA key fingerprint is SHA256:z1Ukk2DxidqKvqEZ+/NW4l78PzRkvQq290SvpfjNlCa.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[192.168.1.129]:2222' (RSA) to the list of known hosts.
root@192.168.1.129's password:
[root@web_server ~]#
test
[root@web_server ~]#
```

### 3.5 Fuzz testing 特征

Fuzz testing (模糊测试) 本是一种安全测试的方法, 通过产生随机的数据输入测试系统查看系统响应或者状态, 以此发现潜在的安全漏洞。部分蜜罐借用 Fuzz testing 的思想实现了蜜罐系统, 该系统的特征如下:

- 响应任意端口的 TCP SYN Packet。
- 根据协议特征, 永远返回正确的响应。
- 返回预定义或者随机的 Payload 特征库集合。

```
$ sudo nmap -sS -Pn -Pn
Host discovery disabled (-Pn). All addresses will
Starting Nmap 7.91 ( https://nmap.org ) at 2021-
Stats: 0:01:39 elapsed; 0 hosts completed (1 up)
SYN Stealth Scan Timing: About 79.27% done; ETC:
Stats: 0:01:40 elapsed; 0 hosts completed (1 up)
SYN Stealth Scan Timing: About 79.72% done; ETC:
Nmap scan report for
Host is up (1.3s latency).
```

PORT	STATE	SERVICE
1/tcp	open	tcpmux
3/tcp	open	compressnet
4/tcp	open	unknown
6/tcp	open	unknown
7/tcp	open	echo
9/tcp	open	discard
13/tcp	open	daytime
17/tcp	open	qotd
19/tcp	open	chargen
20/tcp	open	ftp-data
21/tcp	open	ftp
22/tcp	open	ssh
23/tcp	open	telnet
24/tcp	open	priv-mail
25/tcp	open	smtp
26/tcp	open	rsftp
30/tcp	open	unknown
32/tcp	open	unknown
33/tcp	open	dsp
37/tcp	open	time
42/tcp	open	nameserver
43/tcp	open	whois
49/tcp	open	tacacs
53/tcp	open	domain
70/tcp	open	gopher
79/tcp	open	finger
80/tcp	open	http

2021-01-04 17:49:19 8080 / upnp

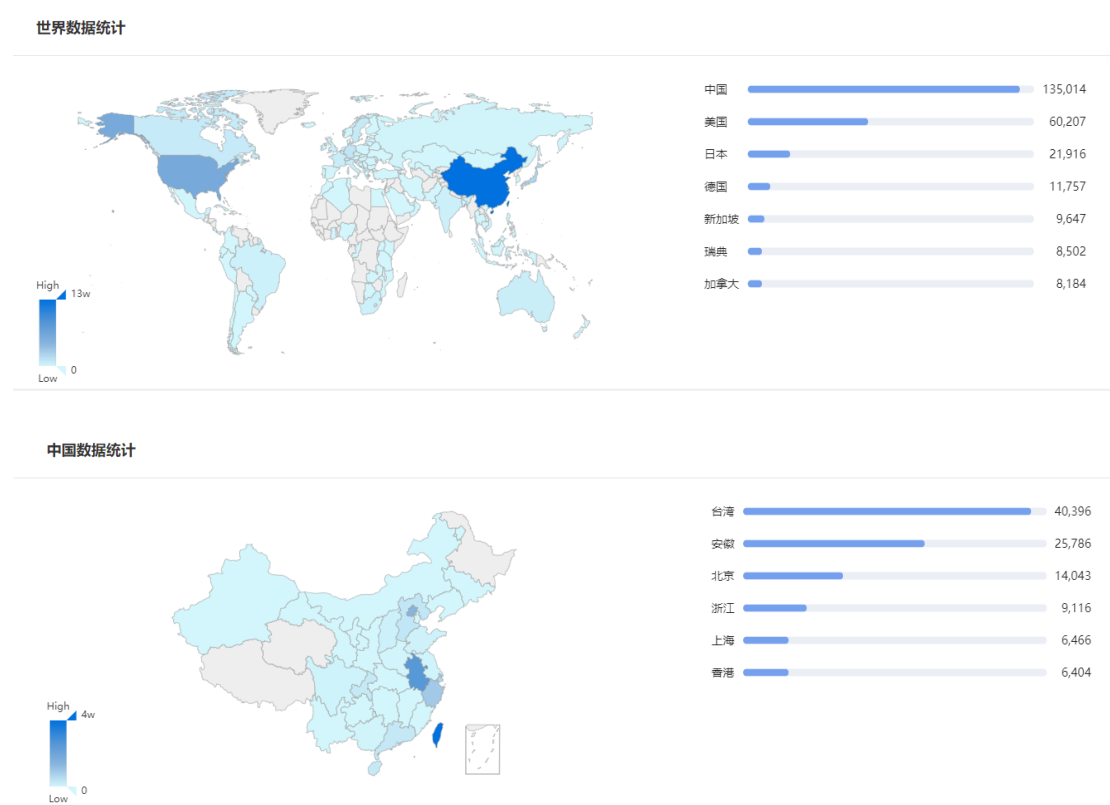
HTTP/1.1 200 OK  
Server: 360 web server, 792/71644 HTTP Server version 2.0 - TELDAT S.A., A10WS/1.00, ADB Broac  
Adaptec ASM 1.1, AirTies/ASP 1.0 UPnP/1.0 miniupnpd/1.0, Allegro-Software-RomPager/4.06, Ami  
-Coyote/1.1, Apache/2.2.15 (CentOS), Apache/2.4.29 (Ubuntu), Apache/2.4.6 (Red Hat Enterprise Li  
L/1.0.2k-fips, App-webs/, ArGoSoft Mail Server Pro for WinNT/2000/XP, Version 1.8 (1.8.9.4), Avigil  
HTTP/1.1, Boa/0.93.15, Boa/0.94.13, Boa/0.94.14rc20, Boa/0.94.14rc21, Boa/0.94.7, BolidXMLRPC/1  
1, CPWS, CVM, Caddy, Cam, Cambium HTTP Server, Camera Web Server, CentOS WebPanel: Prote  
CherryPy/2.3.0, CherryPy/3.1.0beta3 WSGI Server, CherryPy/8.1.2, CirCarLife Scada v4.2.3, Cirpark  
Control4 Web Server, CouchDB/1.6.1 (Erlang OTP/18), CouchDB/1.6.1 (Erlang OTP/R16B03), Couch  
ross Web Server, D-Link Web Server 0.01, DNVRs-Webs, DVR-HttpServer/1.0, DVRDVS-Webs, DW:  
ffice, Destiny, DpmtspKarawangkab\_HTTP\_SERVER, E2EE Server 1.0, EBox, EShare Http Server/1.0,  
Delta Networks Inc., Embedthis-Appweb/3.2.3, Embedthis-Appweb/3.3.1, Embedthis-http, Entrust,  
SU ServerView iRMC S4 Webserver, FileMakerPro/6.0Fv4 WebCompanion/6.0v3, Flussonic, GSHD/  
Open Source Edition 4.0, GoAhead-Webs, GoAhead-Webs/2.5.0, GoAhead-http, GoTTY, H3C-Min  
Server, HTTP Server 1.0, HTTP Software 1.1, HTTPD, HTTPD Web Server, HTTPD-HR Server powerre  
nver built fo SMKN 1 Kaligondang, Http Server, Httpd, Httpd/1.0, Hydra/0.1.8, IBM HTTP Server, IIS  
a Logo, IPOffice/ IceWarp/12.1.1.4 x64, IceWarp/9.4.2, IdeaWebServer/0.83.292, If you want know,  
1.0, JAWS/1.0 Jan 21 2017, JBoss-EAP/7, JdVR/4.0, JFinal 4.5, JWS, Jetty(6.1.19), KMS\_ACCESS, Keil  
E Router Webs, Lanswitch - V100R003 HttpServer 1.1, Linux, HTTP/1.1, DIR-860L Ver 1.01, Linux/2.0  
eMediaCenter/1.0, Linux/3.10.104 eHomeMediaCenter/1.0, Linux/3.10.33 UPnP/1.0 Teleal-Cling/1.0  
-Domino, MIPS LINUX/2.4 UPnP/1.0 miniupnpd/1.0, MJP-G-Streamer/0.2, MS-SDK-HttpServer/1.0, I  
-Appweb/12.5.0, Mbedthis-Appweb/2.4.0, Mbedthis-Appweb/2.4.2, Microsoft-HTTPAPI/1.0, Micro:  
ft-IIS/6.0, Microsoft-IIS/7.0, Microsoft-IIS/7.5, Microsoft-IIS/8.0, Microsoft-IIS/8.5, Microsoft-NetCo  
i Embedded Web Server, Mini web server 1, Mini web server 1.0 ZTE corp 2005., Mini web server 1  
u qualds got a smint?), MonitorServer/0.10.5.363 Python/2.7.5, Monitorix HTTP Server, Monkey, M  
NVR Webserver, Net-OS 5.xx UPnP/1.0, NetBox Version 2.8 Build 4128, NetEVI/3.10, Netwave IP C  
13.0-01 (OSS), Nexus/3.9.0-01 (OSS), Nginx, Nginx Microsoft-HTTPAPI/2.0, Nucleus/4.3 UPnP/1.0 v  
OpenBSD httpd, Oracle Containers for J2EE, Oracle GlassFish Server 3.1.2.2, Oracle XML DB/Oracle  
-Application-Server-11g, Oracle-HTTP-Server, Oracle-HTTP-Server-11g, Oracle\_WebDb\_Listener/2.

该蜜罐很容易通过人工判断, 其目的为模拟蜜罐 fuzzing 特征, 通过预定义大量的关键字实现对扫描器的干扰。该类蜜罐可以通过跨服务的特征进行判断, 如开放了 HTTP 服务同时响应了 upnp 协议, 或者根据 server 的长度或者个数来判断。由于未知哪种蜜罐产品提供的这个蜜罐服务, quake 将此蜜罐标记为未知蜜罐, 可以使用语法 app:”未知蜜罐”搜索。

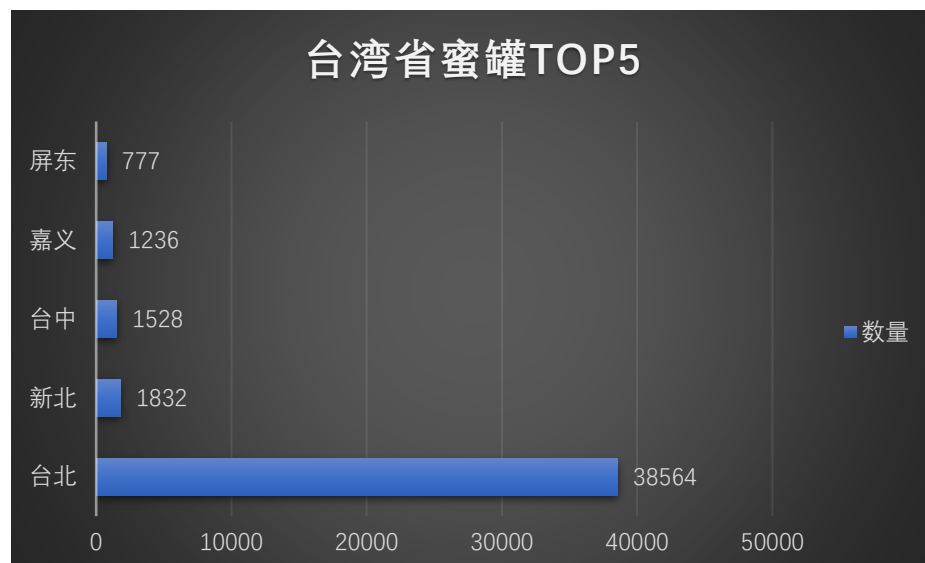
## 4. 开源蜜罐的使用情况

### 4.1 蜜罐分布

在确定了部分开源蜜罐的特征后，我们利用特征进行了全网匹配，发现了 369161 条服务数据和 72948 个独立 ip。全球和全国蜜罐分布如图所示。



可以看到在这些开源蜜罐中，中国的数量是最多的。其中，中国台湾占据了 1/3，位于国内第一。并且在全球省份排名中，台湾省的数量是第一的。



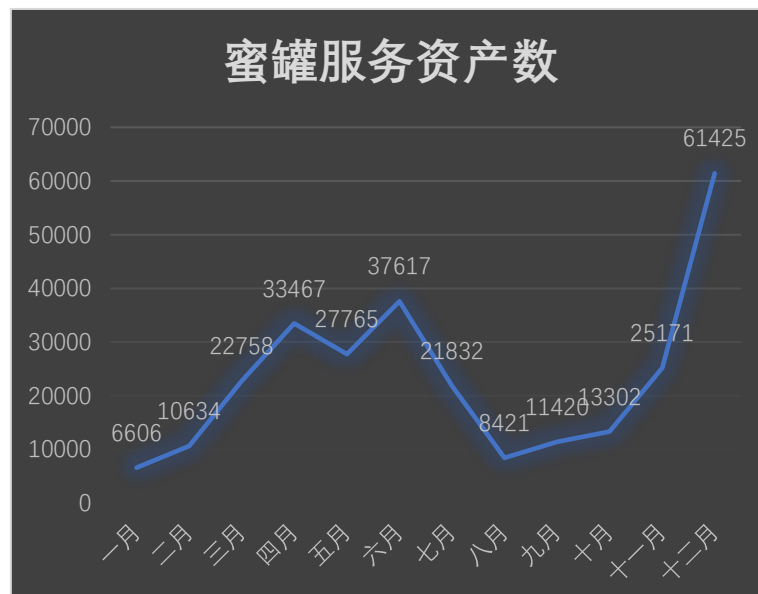
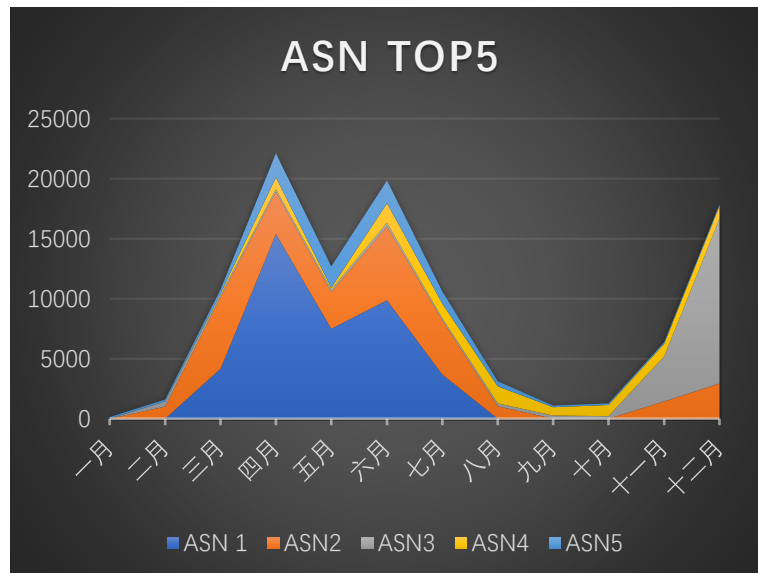


从 ASN 的分布上来看，ASN 数量全球 TOP5 如表所示。发现开源蜜罐主要还是部分部署在云厂商或者教育网中。

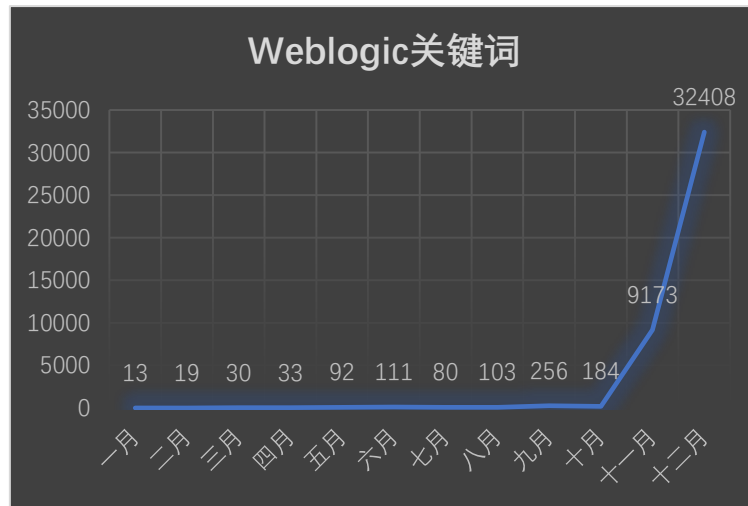
ASN	数量	类型
ASN1	40604	云厂商 (hostus)
ASN2	29649	教育网
ASN3	26926	云厂商 (Amazon)
ASN4	17873	教育网
ASN5	16713	云厂商 (Linode)

## 4.2 生命周期

结合蜜罐服务资产数和 ASNTOP5 全年的分布可以看，蜜罐数量在全年由三个峰值，分别为四月、六月和十二月。



在之前讨论蜜罐的 fuzz testing 时，发现在响应中含有大量与服务有关的关键词，用来干扰扫描器服务识别。其中我发现在服务的响应中含有 weblogic 关键词的蜜罐在十一月开始爆发，我们知道在十月份 CVE-2020-14882 weblogic 未授权命令执行漏洞被披露出来。我们可以做一个猜想，此类蜜罐可以根据热点漏洞进行灵活配置，以达到捕捉扫描器的目的。



## 5. 结论

本文通过通过蜜罐协议返回特征、协议实现的缺陷、明显的 WEB 特征和 Fuzz testing 的特征对常见的 20 种开源蜜罐进行了分析。我们的研究发现，互联网中存在有超过 369161 个蜜罐服务，这些蜜罐都可以通过最简单的特征被检测出来，因为这些蜜罐都是在默认配置情况下被开放在互联网上，基本上是一种自我暴露的状态。从全球分布上来看中国台湾集中了大量的蜜罐，在全球蜜罐的 ASN 分布中，主要集中在云厂商和教育网络中。同时在全年的蜜罐数量上在四月份、六月份和十二月份存在三个峰值，并且通过部分蜜罐响应的关键字来看，蜜罐的数量可能会随着热点漏洞的披露而增长。

最后，本文种所涉及的蜜罐均可在 Quake 种搜素，搜素语法见附录。或者可以通过 Quake 命令行工具进行识别。

```
[kali@kali:~]-[09:57:03 PM]
->$ quake honeypot 13. 221
[+] Search with 13. 221
[!] Looks like a weblogic honeypot system!
[kali@kali:~]-[09:57:12 PM]
->$ quake honeypot 13. 222
[+] Search with 13. 222
[+] Looks like a real system!
[kali@kali:~]-[09:57:23 PM]
->$
```

## 6. 参考

- [1] 蜜罐技术研究新进展[J]. 石乐义,李阳,马猛飞. 电子与信息学报. 2019(02)
- [2] 基于数据包分片的工控蜜罐识别方法[J]. 游建舟,张悦阳,吕世超,陈新,尹丽波,孙利民. 信息安全学报. 2019(03)
- [3] VETTERL, A., AND CLAYTON, R. Bitter harvest: Systematically fingerprinting low- and medium-interaction honeypots at internet scale. In 12th USENIX Workshop on Offensive Technologies, WOOT' 18.
- [4] <http://books.gigatux.nl/mirror/honeypot/final/ch09lev1sec1.html>
- [5] [https://mp.weixin.qq.com/s/\\_hpJP6bTuoH-3cQtDawG0w](https://mp.weixin.qq.com/s/_hpJP6bTuoH-3cQtDawG0w)
- [6] <https://www.avira.com/en/blog/new-mirai-variant-aisuru-detects-cowrie-opensource-honeypots>
- [7] <https://hal.archives-ouvertes.fr/hal-00762596/document>
- [8] <https://subs.emis.de/LNI/Proceedings/Proceedings170/177.pdf>
- [9] <https://www.freebuf.com/articles/ics-articles/230402.html>
- [10] <http://russiansecurity.expert/2016/04/20/mysql-connect-file-read/>
- [11] <https://github.com/mushorg/conpot>
- [12] <https://github.com/cowrie/cowrie>
- [13] <https://github.com/DinoTools/dionaea>
- [14] <https://github.com/jordan-wright/elasticchoney>
- [15] <https://github.com/bontchev/elasticpot>
- [16] <https://github.com/mushorg/glastopf>
- [17] <https://github.com/hacklcx/HFish/>
- [18] <https://github.com/omererdem/honeything>
- [19] <https://github.com/desaster/kippo>
- [20] <https://github.com/madirish/kojoney2>
- [21] <https://github.com/jrwren/nepenthes>
- [22] <https://github.com/thinkst/opencanary>
- [23] <https://github.com/Gifts/Rogue-MySQL-Server>
- [24] <https://github.com/jaksi/sshesame>
- [25] [https://github.com/Cymmetria/weblogic\\_honeypot](https://github.com/Cymmetria/weblogic_honeypot)
- [26] <https://github.com/bg6cq/whoisscanme>
- [27] <https://github.com/zeroq/amun>
- [28] <https://github.com/foospidy/HoneyPy>
- [29] <https://github.com/Cymmetria/StrutsHoneypot>

## 附录

蜜罐	QUAKE DORK
STRUTSHONEYPOT	app:"StrutsHoneypot"
CONPOT HTTP 蜜罐	app:"Conpot Http 蜜罐"
CONPOT MODBUS 蜜罐	app:"Conpot modbus 蜜罐"
CONPOT S7 蜜罐	app:"Conpot s7 蜜罐"
KIPPO 蜜罐	app:"kippo 蜜罐"
HONEYPY HTTP 蜜罐	app:"Honey.py Http 蜜罐"
HONEYPY ES 蜜罐	app:"Honey.py ES 蜜罐"
AMUN IMAP 蜜罐	app:"amun imap 蜜罐"
AMUN HTTP 蜜罐	app:"amun http 蜜罐"
NEPENTHES NETBIOS 蜜罐	app:"Nepenthes netbios 蜜罐"
NEPENTHES FTP 蜜罐	app:"Nepenthes FTP 蜜罐"
SSHESAME SSH 蜜罐	app:"sshesame ssh 蜜罐"
OPENCANARY 蜜罐管理后台	app:"opencanary 蜜罐管理后台"
DIONAEA SIPD 蜜罐	app:"Dionaea sipd 蜜罐"
DIONAEA SMBD 蜜罐	app:"Dionaea smbd 蜜罐"
DIONAEA HTTP 蜜罐	app:"Dionaea Http 蜜罐"
DIONAEA MSSQL 蜜罐	app:"Dionaea MSSQL 蜜罐"
DIONAEA FTP 蜜罐	app:"Dionaea ftp 蜜罐"
DIONAEA MEMCACHED 蜜罐	app:"Dionaea Memcached 蜜罐"
KOJONEY SSH 蜜罐	app:"Kojoney SSH 蜜罐"
WEBLOGIC 蜜罐	app:"weblogic 蜜罐"
MYSQL 蜜罐	app:"MySQL 蜜罐"
HFISH 蜜罐	app:"HFish 蜜罐"
HFISH 蜜罐管理后台	app:"HFish 蜜罐管理后台"
HONEYTHING 物联网蜜罐	app:"honeything 物联网蜜罐"
ELASTICSEARCH 蜜罐	app:"elasticsearch 蜜罐"
HOSTUS 蜜罐	app:"HostUS 蜜罐"
WHOISSCANME 蜜罐	app:"whoisscanme 蜜罐"
未知蜜罐	app:"未知蜜罐"
COWRIE TELNETD 蜜罐	app:"Cowrie telnetd 蜜罐"
GLASTOPF 蜜罐	app:"glastopf 蜜罐"