

0x01 赛题要求

本题中赛事方提供了某一失陷主机在某段时间内的通信流量包，要求选手通过流量审计、追踪溯源等方式找出控制该失陷主机的C2服务器

恶意流量分析(阶段一)

设计说明

恶意软件通常会对其通讯进行加密传输甚至模拟一些正常软件的特征从而避免直接检测，而正常软件在使用过程中也同样会产生大量的加密流量，二者混在一起便会导致检测准确率降低，使得真正的恶意流量无法有效被检测。本题文件中提供了一段时间的失陷主机的通讯流量，参赛选手需审计该流量包，进行 **追踪溯源**，找到真正的恶意服务器，最终获取key，此外本题还设有可以获得额外分数的彩蛋题目。

数据说明

文件: `stage1`
密码: `datacon2021stage1`
md5: `5858bfd9f63db6d33facef6f99bba`

提交形式和文件格式

直接提交获取到的key和彩蛋key。
key为16位随机字符串，该字符串前有显著提示，告知选手该字符串为key
彩蛋为32位随机字符串，该字符串前有显著提示，告知选手该字符串为彩蛋。

提交规则

选手在本题的持续时间内，提交次数不限。
本阶段包含一个彩蛋，彩蛋提供额外分数。

0x02 题目背景及思路

根据给出的hint，得知与失陷主机通信的是**传说中的红队利器Cobalt Strike**的team server。CS的特点是Malleable C2，即“可定制”的C2流量，可以通过配置文件，来修改CS beacon和C2的流量和行为特征，使C2流量混合在目标环境流量中，伪装为正常应用流量，达到欺骗的作用。

根据题目描述，溯源到C2服务器的目的是拿到flag；另一方面，hint中提及“仅靠被动审计是不行的”。所以我们初步认为需要通过主动探测的方式扫描流量中出现的潜在C2主机，而不是针对流量本身进行识别分类。

根据这些关键词，结合搜索到的资料、文章，我们了解到，team server上的Beacon Listener (Beacon是指CS运行在目标主机上的payload)**存在被主动探测手段发现的可能**，因为Beacon Listener所在的Staging Server会提供Payload Staging功能，分阶段进行payload投递，使得任何人都可以通过正确的stage的Url下载stage。只要存储着Beacon配置和payload的stage服务器暴露在公网上，就可以通过主动扫描发现它。

0x03 具体流程

1.主动扫描

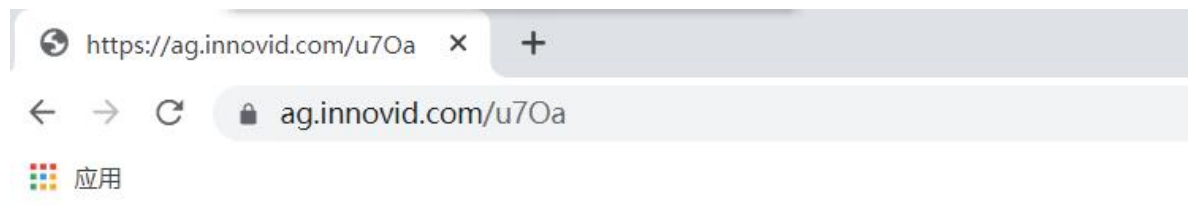
将pcap文件中所有tls会话的handshake阶段的server name提取出来(数字是出现次数):

```
1440 code.jquery.com
372 www.baidu.com
360 jquery.com
186 www.w3school.com.cn
180 www.runoob.com
...
1 12www-mirror-co-uk.amp.cloudflare.com
1 11-www-mirror-co-uk.amp.cloudflare.com
1 0-gravatar-com.amp.cloudflare.com
```

首先常规思路筛选出Alexa top1M以外的域名，**企图缩小范围**。

```
186 www.w3school.com.cn
10 mat1.gting.com
8 wordpress.com
...
1 bizup.cc.danuoyi.tbcache.com
1 ag.innovid.com
1 adm.leju.com
```

根据stage url的生成规则，扫描时需要**额外拼接上url的checksum8校验码**，否则会返回404。根据大佬们公开的逆向出的stage url生成规则，可以得到校验码的生成方法，并最终生成32位或64位payload的4位校验码。



A client error occurred: Resource not found by Assets controller

最终发现并没有什么用，无法下载到任何文件。这说明**攻击者确实使用了CS的Malleable C2功能进行了域名伪造，不能简单粗暴地进行Alexa筛选**。

对所有域名进行扫描，发现如下几个域名可以得到stage文件：

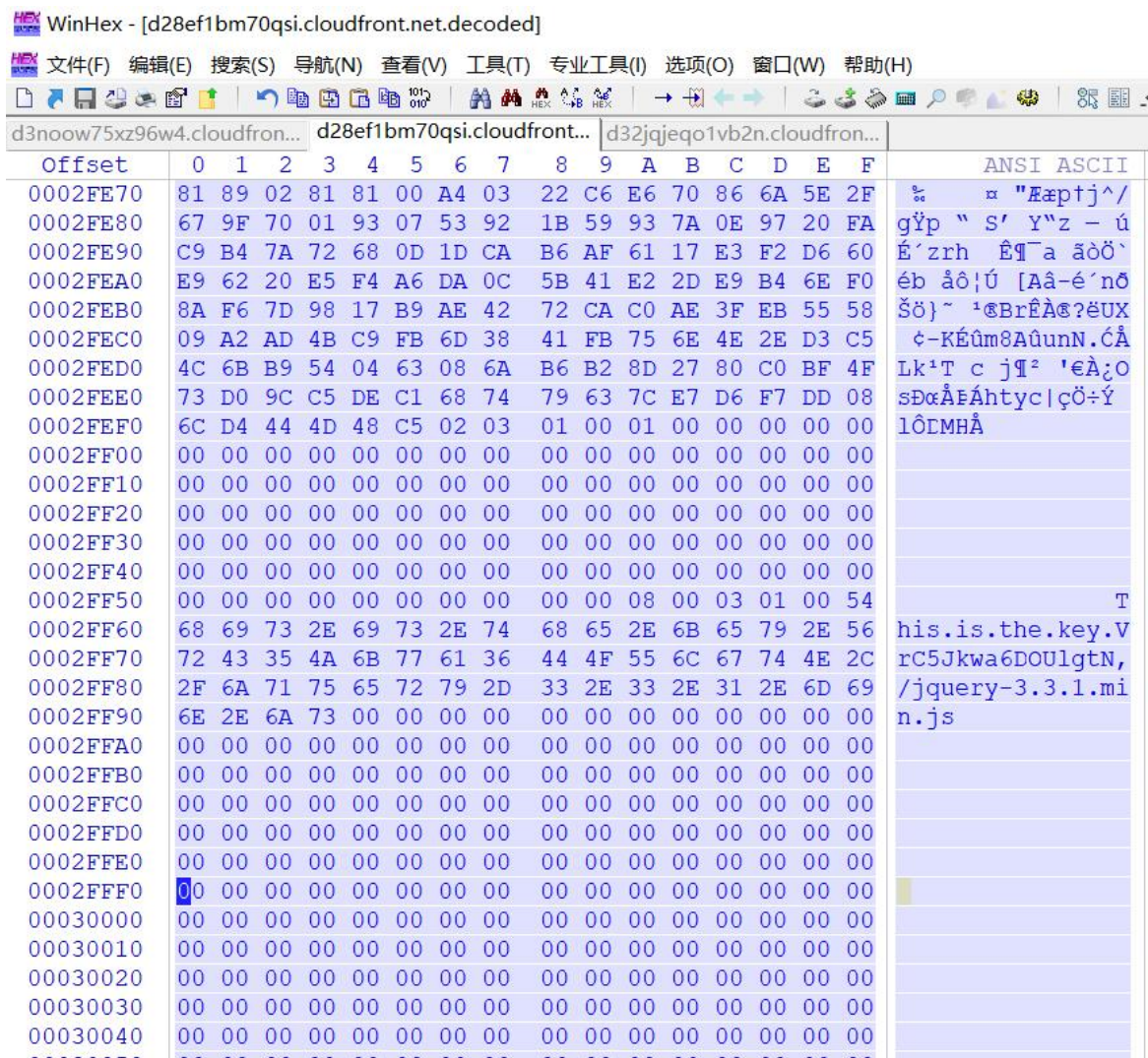
```
27 d28ef1bm70qsi.cloudfront.net
13 d1yr5tm734gilr.cloudfront.net
10 dku6bh98adktv.cloudfront.net
9 d21j8kjjwt8rn6.cloudfront.net
8 d3noow75xz96w4.cloudfront.net
8 d2og948cy5xutu.cloudfront.net
6 d32jqjeqo1vb2n.cloudfront.net
```

2.解析stage配置文件

获取配置文件后，使用如下脚本进行解密：

```
import sys
import struct

filename = sys.argv[1]
data = open(filename, 'rb').read()
t = bytearray(data[0x45:])
(a,b) = struct.unpack_from('<II', t)
key = a
```



WinHex - [d3noow75xz96w4.cloudfront.net.decoded]

文件(F) 编辑(E) 搜索(S) 导航(N) 查看(V) 工具(T) 专业工具(I) 选项(O) 窗口(W) 帮助(H)

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	ANSI	ASCII
0002FEA0	B6	5A	DB	28	78	F3	3F	5A	03	FF	50	10	B2	3E	84	2A	QZ0(xó?Z ŸP ^>,,*	
0002FEB0	51	0C	14	82	AD	6A	42	F1	E7	E5	72	6E	B3	18	13	E7	Q , -jBñçãrn° ç	
0002FEC0	43	76	40	ED	78	79	95	5F	40	1E	17	2C	34	D3	51	72	Cv@ixy• @ ,4óQr	C
0002FED0	41	59	6D	D4	1F	8E	48	D3	D1	B1	C2	88	E6	C8	75	2F	AymÔ ŽHŌÑ±Â^æEu/	
0002FEE0	F6	5D	C2	7A	CC	CB	A4	BA	9C	D6	D0	E4	DE	61	96	CE	øjÂziĚ°œÖĐăPa-î	
0002FEF0	A4	DA	48	0D	3B	99	D0	ED	02	03	01	00	01	00	00	00	uŪH ;™Đí	
0002FF00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0002FF10	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0002FF20	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0002FF30	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0002FF40	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0002FF50	00	00	00	00	00	00	00	00	00	00	00	00	00	08	00	03	01	
0002FF60	00	74	68	69	73	2E	69	73	2E	74	68	65	2E	66	61	6B	this.is.the.fak	
0002FF70	65	2E	63	32	2E	58	44	2C	2F	6A	71	75	65	72	79	2D	e.c2.XD,/jquery-	e
0002FF80	33	2E	33	2E	31	2E	6D	69	6E	2E	6A	73	00	00	00	00	3.3.1.min.js	3
0002FF90	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0002FFA0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0002FFB0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0002FFC0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0002FFD0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0002FFE0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0002FFF0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
00030000	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		

0x04彩蛋

关于彩蛋，唯一的线索就在未知异或密钥的那个文件中，可以观察到，不论是fake文件还是有key的文件，解密后的明文位置都是0x2FF60，说明出题人还是手下留情了。如果参照相关的博客，在最后一步利用脚本从配置文件解析明文信息，而不是用winhex进行异或运算，就无法看出这一点。

可以发现，所有文件在解密为明文后，0x2FF60位置周围都是用于填充的00，因此异或加密之后，相应的位置显示的就是异或密钥本身(00 xor A ==A). 而这一系列配置文件的格式大概率都是相同的，所以在d32jqjeqo1vb2n.cloudfront.net的配置文件中，相应位置的明文也必然是用于填充的00，观察decode后的文件，可以得出异或密钥为0x28的结论：

WinHex - [d32jqjeqo1vb2n.cloudfront.net.decoded]

文件(F) 编辑(E) 搜索(S) 导航(N) 查看(V) 工具(T) 专业工具(I) 选项(O) 窗口(W) 帮助(H)

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	ANSI	ASCII
0002FEB0	60	AA	67	C4	CE	7F	E6	EA	68	C5	66	D7	2C	53	8A	12	*qĀî æêhĀf»,SŠ	
0002FEC0	7A	A9	30	C2	AB	D4	AF	E9	4A	DB	EF	2F	91	DF	AA	41	z@0Ā«Ĉ-éJŮî/'ß*A	z□
0002FED0	05	AE	C1	86	AA	17	3D	BC	CE	CA	12	02	FF	8D	4B	25	@Ā†«=¼ĪĀ Ÿ K%	□□
0002FEE0	52	DF	C4	DF	19	7B	4E	F6	0B	66	2A	58	FB	5C	07	59	RßĀß {Nö f*xŬ\ Y	R□
0002FEF0	E4	41	11	4E	A0	A7	D9	82	9A	88	CC	AD	FF	48	77	F1	äA N ŠŮ,š^î-ŸHwñ	□
0002FF00	83	F8	DF	4A	65	2A	2B	29	28	29	28	28	28	28	28	28	føßJe*+)) ((((((□
0002FF10	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28	((((((((((
0002FF20	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28	((((((((((
0002FF30	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28	((((((((((
0002FF40	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28	((((((((((
0002FF50	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28	((((((((((
0002FF60	28	28	28	28	28	28	28	28	28	20	28	2B	29	28	51	47	((((((((((
0002FF70	5D	06	40	49	5E	4D	06	4E	47	5D	46	4C	06	5C	40	4D] @I^M NG]FL \@M]□@I
0002FF80	06	4A	47	46	5D	5B	06	4B	47	46	4F	5A	49	5C	5D	44	JGF][KGFOZI\]D	□JGF
0002FF90	49	5C	41	47	46	5B	06	4E	4E	49	61	58	67	5A	4C	72	I\AGF[NNiAxgZLr	I\AG
0002FFA0	4B	11	5B	44	64	6F	4B	67	7E	5A	71	49	4F	66	1B	6E	K [DdoKg~ZqIOf n	K□[D
0002FFB0	6F	7C	7E	78	4D	5F	43	04	07	42	59	5D	4D	5A	51	05	o ~xM_C BY]MZQ	o ~x
0002FFC0	1B	06	1B	06	19	06	45	41	46	06	42	5B	28	28	28	28	EAF B[(((□□□□
0002FFD0	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28	((((((((((
0002FFE0	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28	((((((((((

WinHex - [d32jqjeqo1vb2n.cloudfront.net.decoded]

文件(F) 编辑(E) 搜索(S) 导航(N) 查看(V) 工具(T) 专业工具(I) 选项(O) 窗口(W) 帮助(H)

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	ANSI	ASCII
0002FEB0	48	82	4F	EC	E6	57	CE	C2	40	ED	4E	FF	04	7B	A2	3A	H,OiæWîÂ@iNÿ {ç:	H□
0002FEC0	52	81	18	EA	83	FC	87	C1	62	F3	C7	07	B9	F7	82	69	R êfû#Âbóç ¹÷,i	R□
0002FED0	2D	86	E9	AE	82	3F	15	94	E6	EC	3A	2A	D7	A5	63	0D	-têç,? "æi:*x¥c	□
0002FEE0	7A	F7	EC	F7	31	53	66	DE	23	4E	02	70	D3	74	2F	71	z÷i÷lSfB#N pót/q	z□
0002FEF0	CC	69	39	66	88	8F	F1	AA	B2	A0	E4	85	D7	60	5F	D9	îi9f^ ñª² ä...x`_û	□
0002FF00	AB	D0	F7	62	4D	02	03	01	00	01	00	00	00	00	00	00	«Ð÷bM	□
0002FF10	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		□□□
0002FF20	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		□□□
0002FF30	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		□□□
0002FF40	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		□□□
0002FF50	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		□□□
0002FF60	00	00	00	00	00	00	00	00	00	08	00	03	01	00	79	6F		yo□□□
0002FF70	75	2E	68	61	76	65	2E	66	6F	75	6E	64	2E	74	68	65	u.have.found.the	u.h
0002FF80	2E	62	6F	6E	75	73	2E	63	6F	6E	67	72	61	74	75	6C	.bonus.congratul	.bo
0002FF90	61	74	69	6F	6E	73	2E	66	66	61	49	70	4F	72	64	5A	ations.ffaIpOrdZ	ati
0002FFA0	63	39	73	6C	4C	47	63	4F	56	72	59	61	67	4E	33	46	c9slLGcOVrYagN3F	c9s
0002FFB0	47	54	56	50	65	77	6B	2C	2F	6A	71	75	65	72	79	2D	GTVPewk,/jquery-	GTV
0002FFC0	33	2E	33	2E	31	2E	6D	69	6E	2E	6A	73	00	00	00	00	3.3.1.min.js	3.3
0002FFD0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		□□□
0002FFE0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		□□□

References

- https://www.sohu.com/a/435908844_750628
- <https://www.freebuf.com/articles/network/273480.html>
- <https://cloud.tencent.com/developer/article/1764340>
- <https://www.52pojie.cn/thread-1396671-1-1.html>
- <https://xz.aliyun.com/t/2796>