踩坑记录 - DNS Beacon - 先知社区

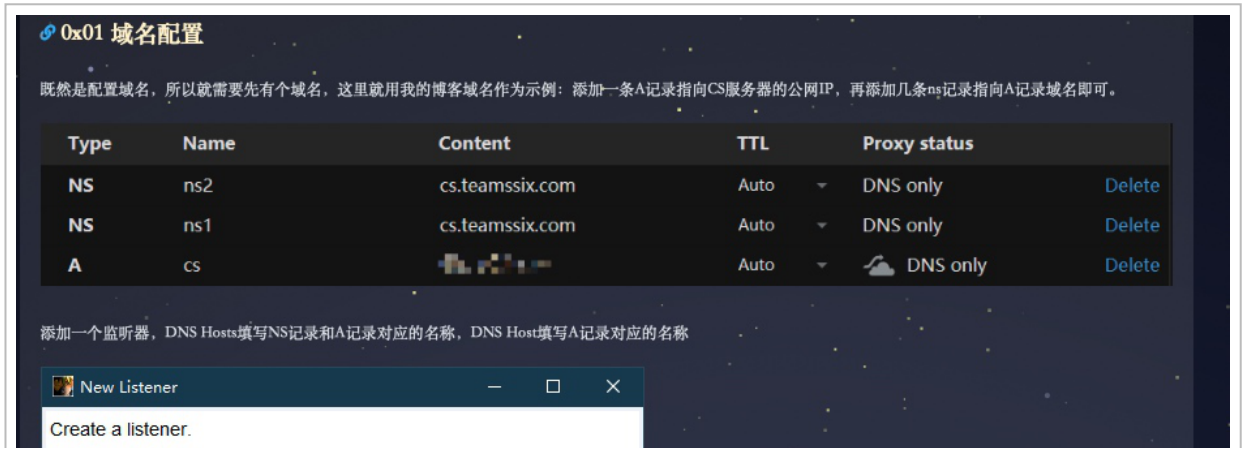
> 先知社区，先知安全技术社区

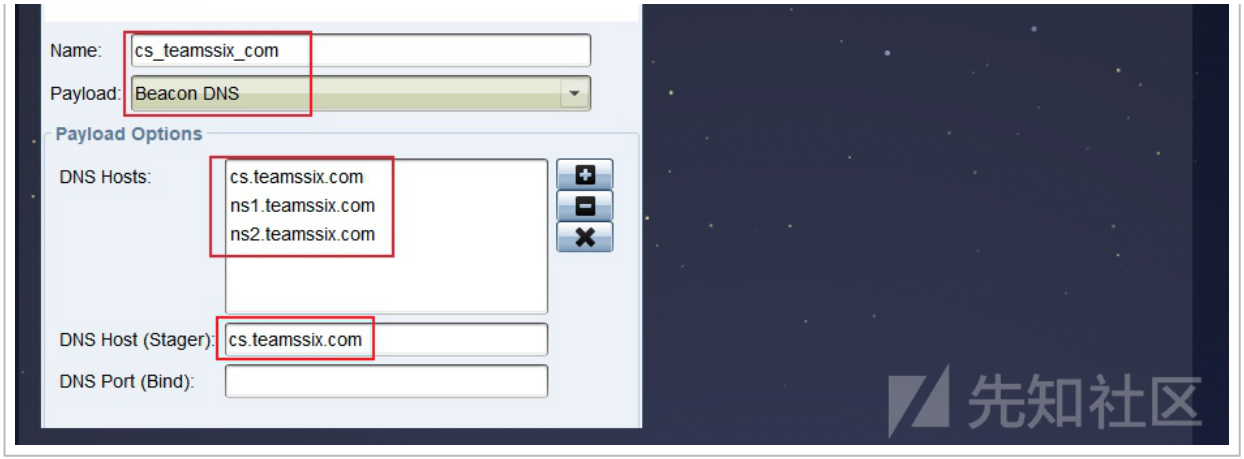
0x00 场景

接触 CS 也不久，之前只用过最基本的一些功能，但近期的比赛中越来越频繁的遇到拿了 Webshell 但发现不出网的机器，所以被迫用上了 DNS Beacon。

0x01 踩坑

网上的同人文基本上都是这样配置：

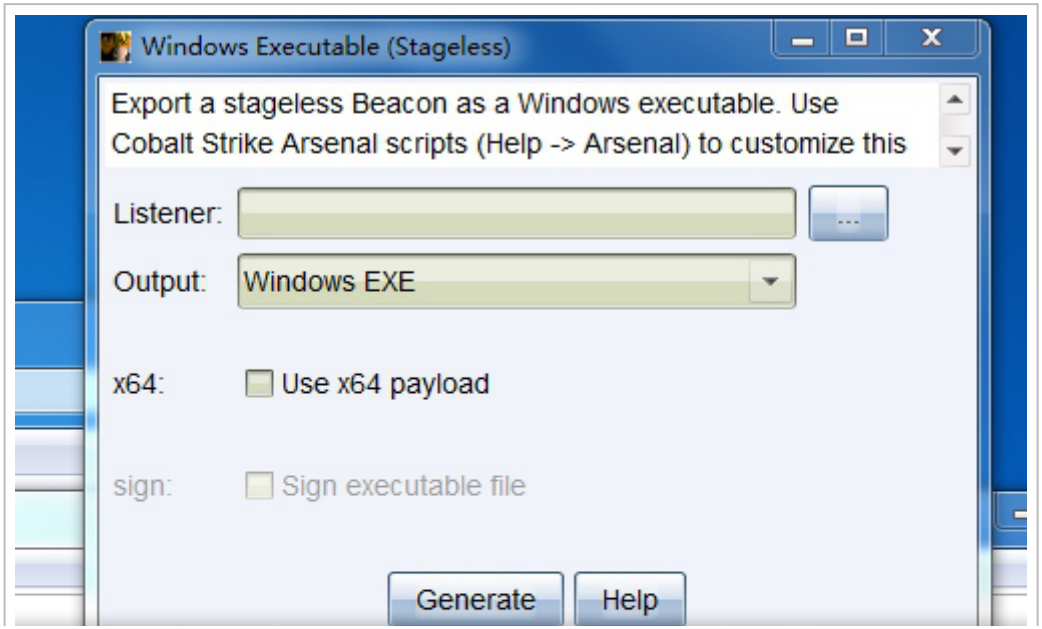




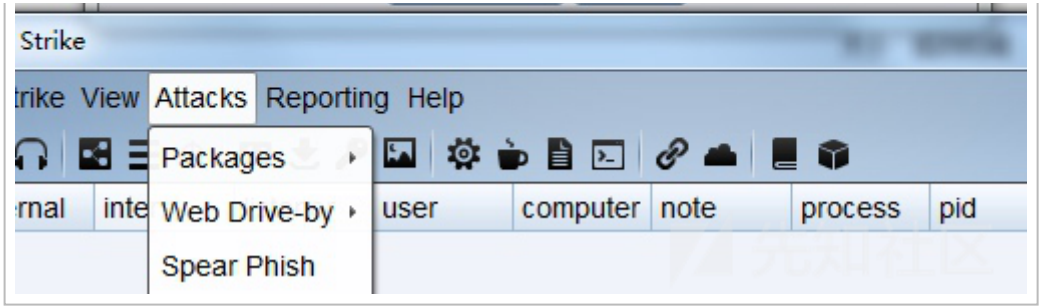
(https://xzfile.aliyuncs.com/media/upload/picture/20200624093146-77c53522-b5ba-1.jpeg)

看起来挺有道理，但跟着走，走着走着发现掉坑了。

这样配置的话，上线确实可以，但成功上线的是下图中这种类型的 beacon(stageless)，文章中却对其没有任何解释，也没有描述不同 beacon 的原理（其实官方英文文档和视频也在含糊其辞）：

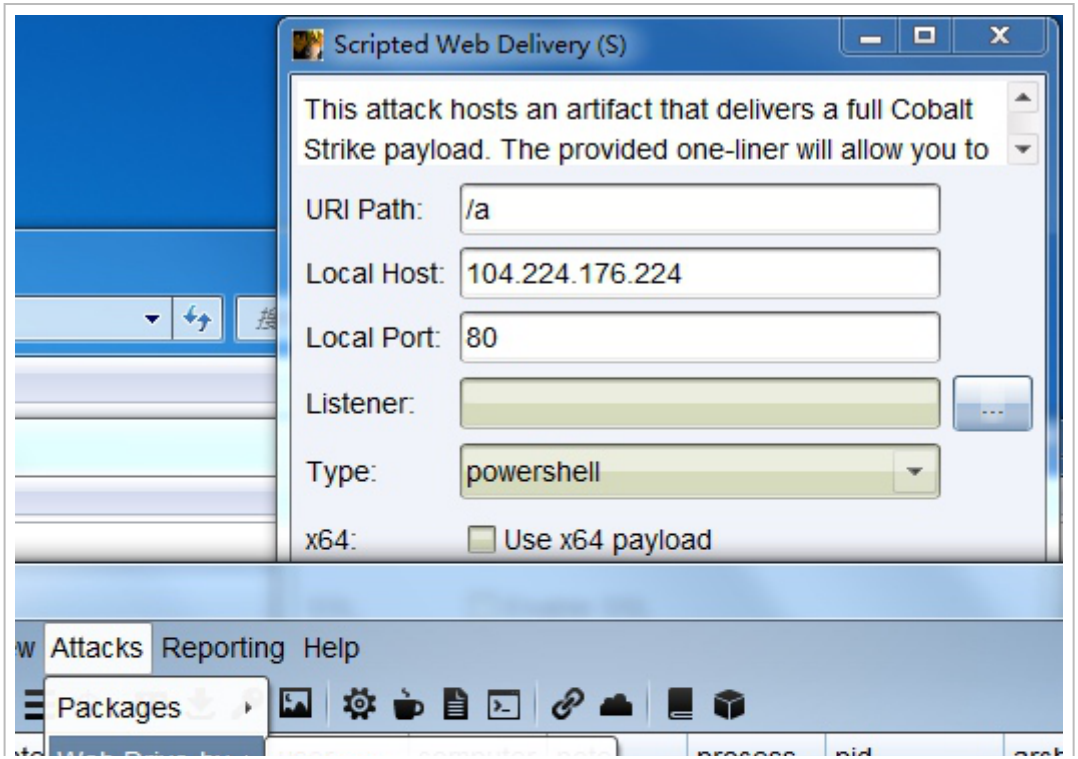


/

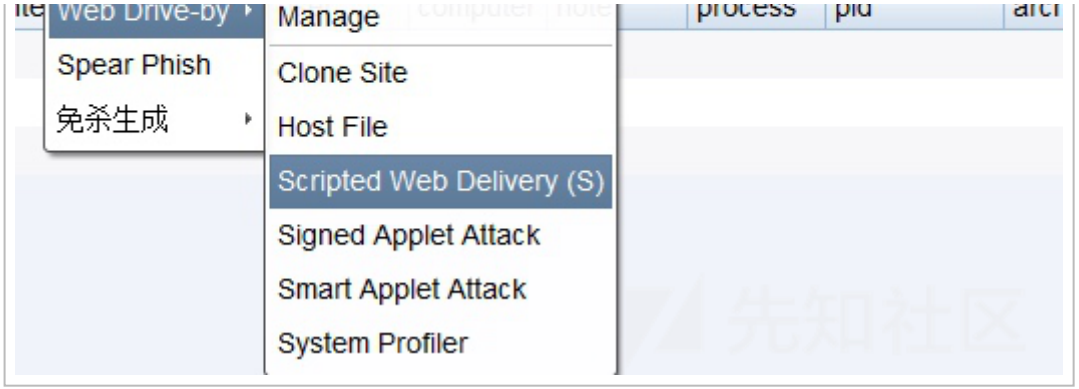


(https://xzfile.aliyuncs.com/media/upload/picture/20200624093224-8e2623ee-b5ba-1.jpeg)

或者同种 PS 版本的：



/

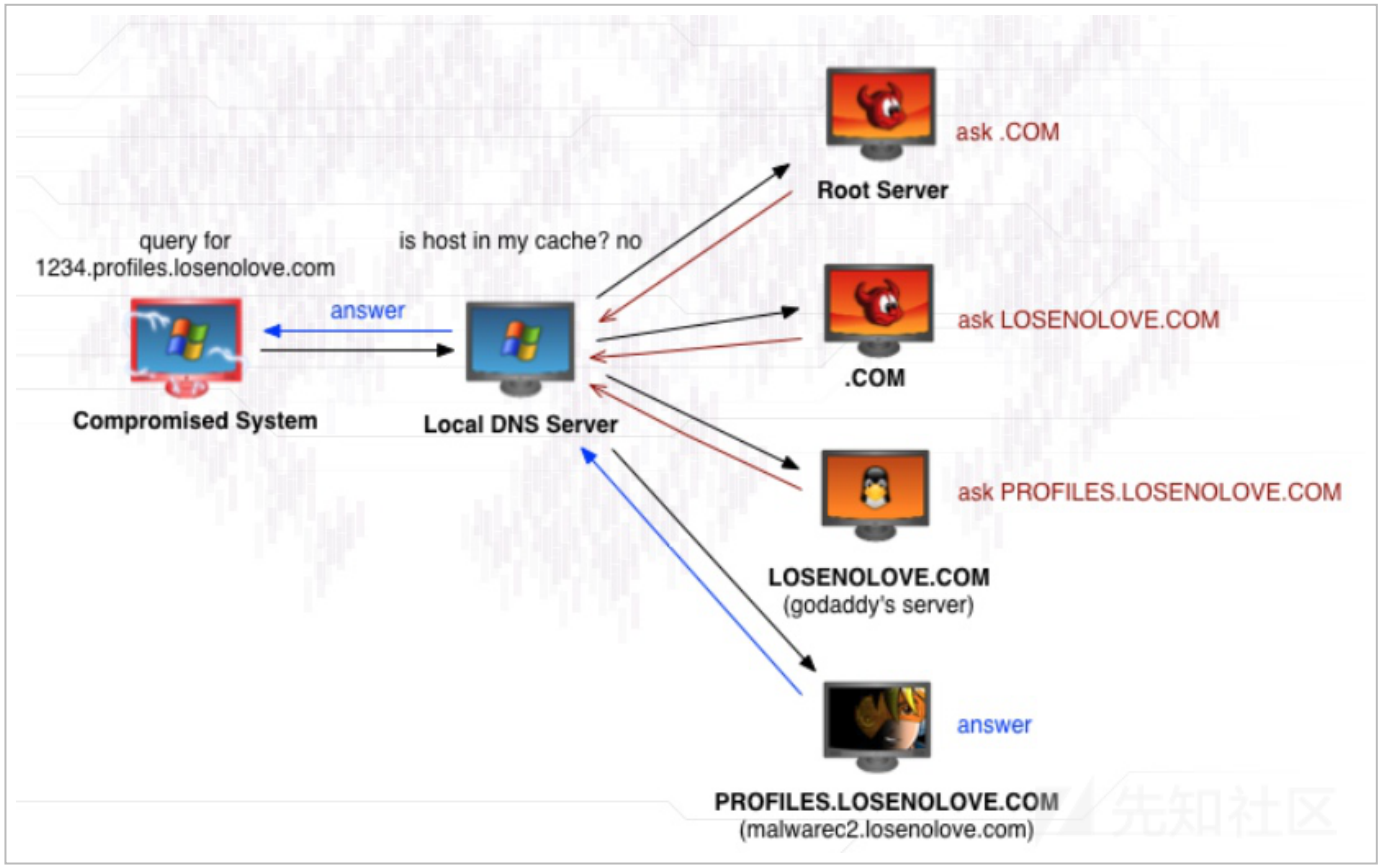


(https://xzfile.aliyuncs.com/media/upload/picture/20200624094303-0b02f36e-b5bc-1.jpeg)

这种 beacon 的特点是 payload 体积庞大，生成的 RAW Payload 也是一个完整 Shellcode Loader 的 PE 文件（约 200KB）而不是一段 shellcode（不足 1KB），虽然经测试也可以将其整体编码后做免杀，但总觉得别扭且麻烦。而且，在内存中解密出一个完整的 PE 文件这种行为也会被特别关照吧。

0x02 Beacon

那么上面看到的两种 beacon 有什么区别呢？查阅官方说明，翻译并归纳总结如下。
CS 的马有两大类：一种是和灰鸽子类似的马，本身是带有完整功能的二进制程序；另一种马短小精悍，也是平时自己 diy 免杀马常用的，分为三个部分——攻击载荷（payload）、传输器（stager）和传输体（stage），传输器通过下载传输体将其拼接成真正可以回连 teamserver 的马子（payload），从而达到比较出色的伪装效果。
上面图中的那类 stageless，顾名思义就是没有 stager 和 stage 的马，而是直接以 payload 为模版生成一个完整的二进制马，当然特征也是最明显的。
而 DNS Beacon 本质上与 HTTP/HTTPS 无异，只是将数据封装进了不同的协议而已。这里借用官网的原图说明下 DNS Beacon 的工作原理：



(https://xzfile.aliyuncs.com/media/upload/picture/20200624094318-13efa36e-b5bc-1.jpeg)

从上图中可以看到，当我们给 CS 的 teamserver 搞了一个域名并配置相应的 A 记录以及指向自身 A 记录的 NS 记录后，DNS 请求就会被迭代查询的本地 DNS 服务器一步一步引向 teamserver，teamserver 收到了服务端的特殊请求后便可以用封装的加密通信协议与之交互了。
道理很简单，可为啥用起来就很坑呢?

0x03 CS4 和前作的区别

DNS Beacon 也是 CS4 中改动比较大的一个，而网上针对它的中文文章确实还比较少，大多数都是 3 代的教程（好好学英语），而配置还是有挺大区别的。
CS4 中的 DNS Beacon 才是真正纯粹的 DNS 隧道，前作中大家常用的方式都是以 http 的形式传递 stage，但这种场景并不是我们要用 DNS 隧道的主要原因，如果你可以走 HTTP 就证明主机可以联网，干嘛放着大路不走，非要走不太稳定的 DNS 呢（有特殊隐蔽需求的除外）？
而现在的 DNS Beacon 则是仅留下了 reverse_dns_txt，所有的 stage 必须都从 DNS 隧道中下载，这也是为什么上面的配置中只有 stageless 的可以回连但碰到需要下载 stage 的就会失败的原因，因为配置本身就有问题呀。

0x04 配置

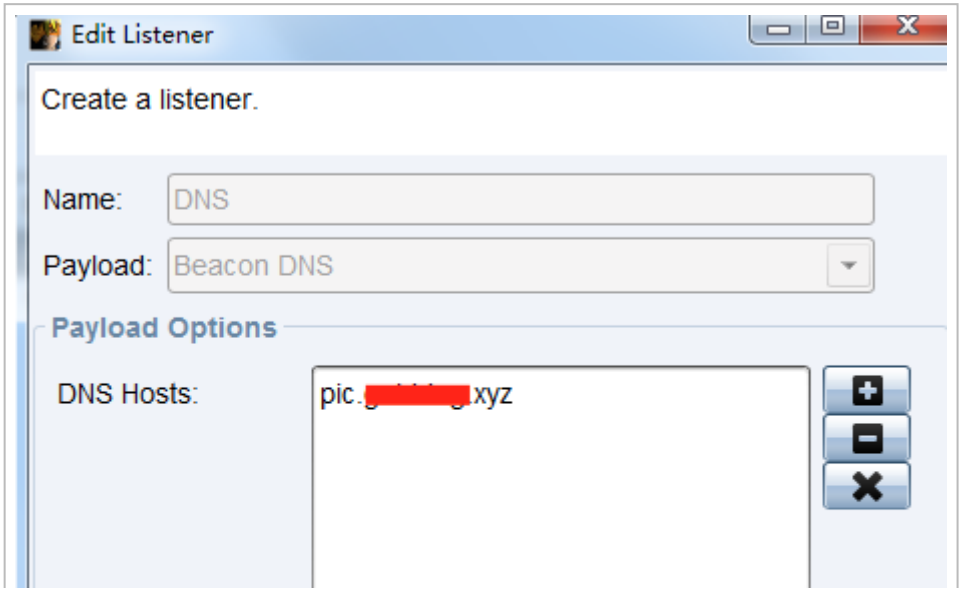
多说无益，直接看看官方文档中是怎么配置的吧（相关域名可对照上面的官方文档原理图）：

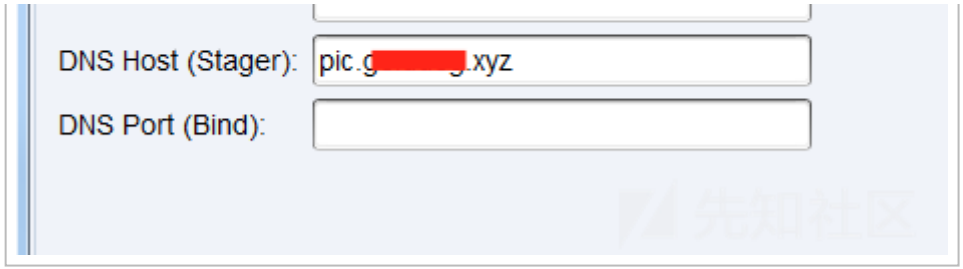

New Listener
Create a listener.
Name:
local - dns
Payload:
Beacon DNS
Payload Options
DNS Hosts:
profiles.losenolove.com
freepics.losenolove.com
game.losenolove.com
DNS Host (Stager):
profiles.losenolove.com
DNS Port (Bind):

/

(https://xzfile.aliyuncs.com/media/upload/picture/20200624094607-78b990d4-b5bc-1.jpeg)

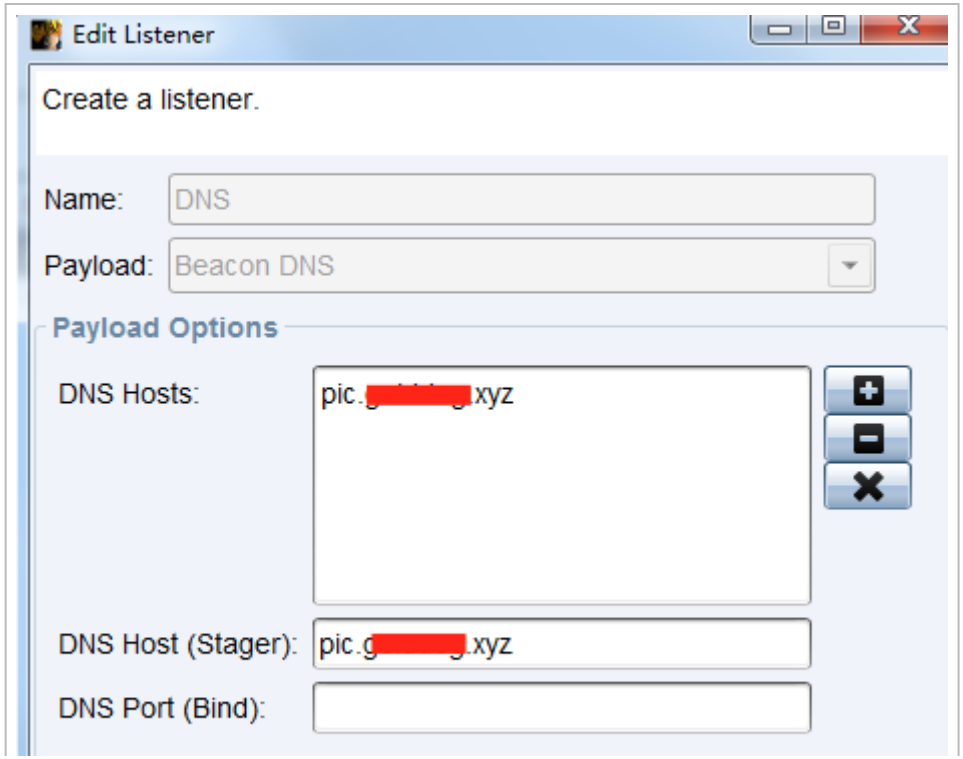
这里一开始没有仔细看，看到了 3 个 DNS HOSTS 就以为和网友帖子配的是一样的，但其实不是。人家真正的 A 记录并没有写在配置里面，这里面的域名都是 NS 记录!
其实仔细想想也知道，A 记录是用来解析 IP 地址的，通信用流量封装在了 DNS 请求中，由上级 DNS 服务器代发，跟 teamserver 的 IP 有毛线关系?
因此在域名解析记录中，配一条 A 记录指向 teamserver 的 IP，再配 1 条或多条 NS 记录用来做隧道，域名指向 A 记录。就像这样：





(https://xzfile.aliyuncs.com/media/upload/picture/20200624094814-c484397e-b5bc-1.png)

在 CS 中配置 HOSTS 为刚刚建立的 NS 记录，配置 stager 域名为其中的一个 NS 记录（如果只配了
一个就写一样的）：



/

(https://xzfile.aliyuncs.com/media/upload/picture/20200624094919-eb7a78fe-b5bc-1.png)

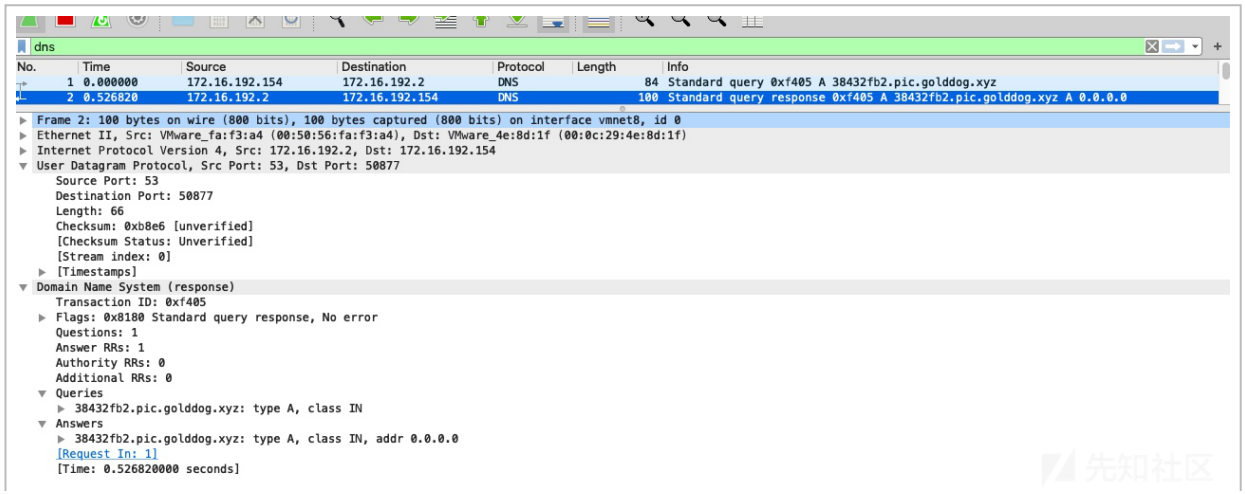
然后就可以生成带 stager 的小马或导出 shellcode 自己做免杀了。

0x05 流量

坑踩过了，原理和配置都搞清楚了，本着科学严谨的态度再做个实验验证下流量。

stageless

首先生成个 stageless 的大马并执行。一开始在 checkin 阶段，会有这样一次 DNS 请求的交互：



/

(https://xzfile.aliyuncs.com/media/upload/picture/20200624094952-fed91ea0-b5bc-1.jpeg)

CS 中随即出现了一个 Ghost Beacon：

(https://xzfile.aliyuncs.com/media/upload/picture/20200624095006-073a1fae-b5bd-1.jpeg)

这里要注意还有个坑，想让它激活还需要在 beacon 里面输入 checkin 命令，不然 DNS 服务器是不会返回激活指令的。一直不 chechin，他就会每过一段时间请求一个新的子域名，返回的“IP” 都是 0.0.0.0。如下图：

No.	Time	Source	Destination	Protocol	Length	Info
181	8.643809	172.16.192.154	172.16.192.2	DNS	83	Standard query 0xd92c A 7f0b6f6.pic.golddog.xyz
182	8.829485	172.16.192.2	172.16.192.154	DNS	99	Standard query response 0xd92c A 7f0b6f6.pic.golddog.xyz A 0.0.0.0
184	37.255596	172.16.192.154	172.16.192.2	DNS	84	Standard query 0x936b A 114ae5f2.pic.golddog.xyz
185	37.259726	172.16.192.2	172.16.192.154	DNS	100	Standard query response 0x936b A 114ae5f2.pic.golddog.xyz A 0.0.0.0
188	68.845424	172.16.192.154	172.16.192.2	DNS	83	Standard query 0x4d6d A 7f0b6f6.pic.golddog.xyz
189	68.850564	172.16.192.2	172.16.192.154	DNS	99	Standard query response 0x4d6d A 7f0b6f6.pic.golddog.xyz A 0.0.0.0
198	97.270300	172.16.192.154	172.16.192.2	DNS	84	Standard query 0xe27d A 114ae5f2.pic.golddog.xyz
199	97.276822	172.16.192.2	172.16.192.154	DNS	100	Standard query response 0xe27d A 114ae5f2.pic.golddog.xyz A 0.0.0.0

(https://xzfile.aliyuncs.com/media/upload/picture/20200624095148-444fd230-b5bd-1.jpeg)

对应的，CS 上面也会出现相同数量的 ghost（也有可能会少几个）。此时在 ghost beacon 中输入 checkin 命令，下一次心跳中，服务端就会返回不一样的 “IP” （类似唤醒指令），例如 checkin 命令对应 0.0.0.243、还有 mode dns 对应 0.0.0.241 等等。不管是什么指令，只要有效就可以激活后门，开始进行通信：

No.	Time	Source	Destination	Protocol	Length	Info
212	128.859823	172.16.192.154	172.16.192.2	DNS	83	Standard query 0xe967 A 7f0b6f6.pic.golddog.xyz
213	128.864992	172.16.192.2	172.16.192.154	DNS	99	Standard query response 0xe967 A 7f0b6f6.pic.golddog.xyz A 0.0.0.0
214	157.283985	172.16.192.154	172.16.192.2	DNS	84	Standard query 0x3cb2 A 114ae5f2.pic.golddog.xyz
215	157.289816	172.16.192.2	172.16.192.154	DNS	100	Standard query response 0x3cb2 A 114ae5f2.pic.golddog.xyz A 0.0.0.243
216	157.290514	172.16.192.154	172.16.192.2	DNS	102	Standard query 0xd4bf A www.180.01fdb4af5.114ae5f2.pic.golddog.xyz

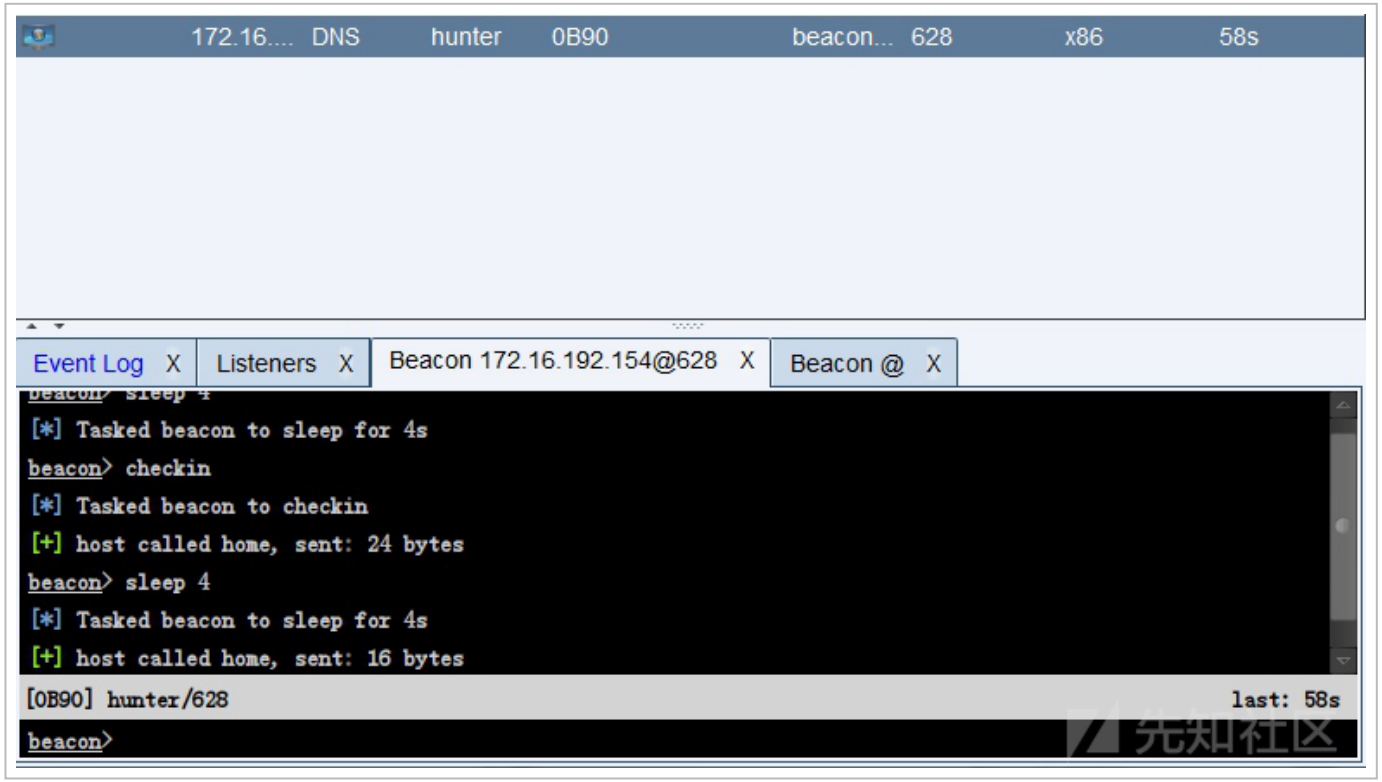
(https://xzfile.aliyuncs.com/media/upload/picture/20200624095203-4d4f9334-b5bd-1.jpeg)

随后的通信便是一些看不懂的子域名：

No.	Time	Source	Destination	Protocol	Length	Info
141	300.614389	172.16.192.2	172.16.192.154	DNS	100	Standard query response 0xe15e A 38432fb2.pic.golddog.xyz A 0.0.0.243
142	300.615039	172.16.192.154	172.16.192.2	DNS	102	Standard query 0x75ab A www.180.05e510709.38432fb2.pic.golddog.xyz
143	300.858184	172.16.192.2	172.16.192.154	DNS	118	Standard query response 0x75ab A www.180.05e510709.38432fb2.pic.golddog.xyz A 0.0.0.0
144	300.858860	172.16.192.154	172.16.192.2	DNS	311	Standard query 0x811d A www.4186a1095e210c755af8d9268ef42c48084797f40.6679c8a5c378ba13caff7f32688
145	301.055805	172.16.192.2	172.16.192.154	DNS	327	Standard query response 0x811d A www.4186a1095e210c755af8d9268ef42c48084797f40.6679c8a5c378ba13ca
146	301.056482	172.16.192.154	172.16.192.2	DNS	148	Standard query 0x5f93 A www.1d2cc3e73517657b5e67bba16026501ac2b4d755785f68d86.25e510709.38432fb2.
147	301.366108	172.16.192.2	172.16.192.154	DNS	164	Standard query response 0x5f93 A www.1d2cc3e73517657b5e67bba16026501ac2b4d755785f68d86.25e510709.
148	301.366860	172.16.192.154	172.16.192.2	DNS	98	Standard query 0xee12 A api.033062a97.38432fb2.pic.golddog.xyz
149	301.557396	172.16.192.2	172.16.192.154	DNS	114	Standard query response 0xee12 A api.033062a97.38432fb2.pic.golddog.xyz A 0.0.0.64

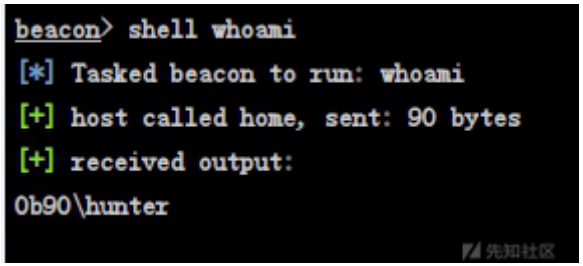
(https://xzfile.aliyuncs.com/media/upload/picture/20200624095220-56f82c52-b5bd-1.jpeg)

子域名中看起来的随机字符串其实就是通过 A 记录中夹带封装的数据，只不过由于数据量很小，只能少量多次发送接收，因此 checkin 的过程需要等一段时间，并不会像 HTTP/HTTPS 那样瞬间上线。
通信过后，beacon 便正常上线了（注意，由于 DNS 数据包的来源是代理查询的 DNS 服务器并不是被控机器的真实出口 IP，因此这里不会显示 external 的 IP 地址）：

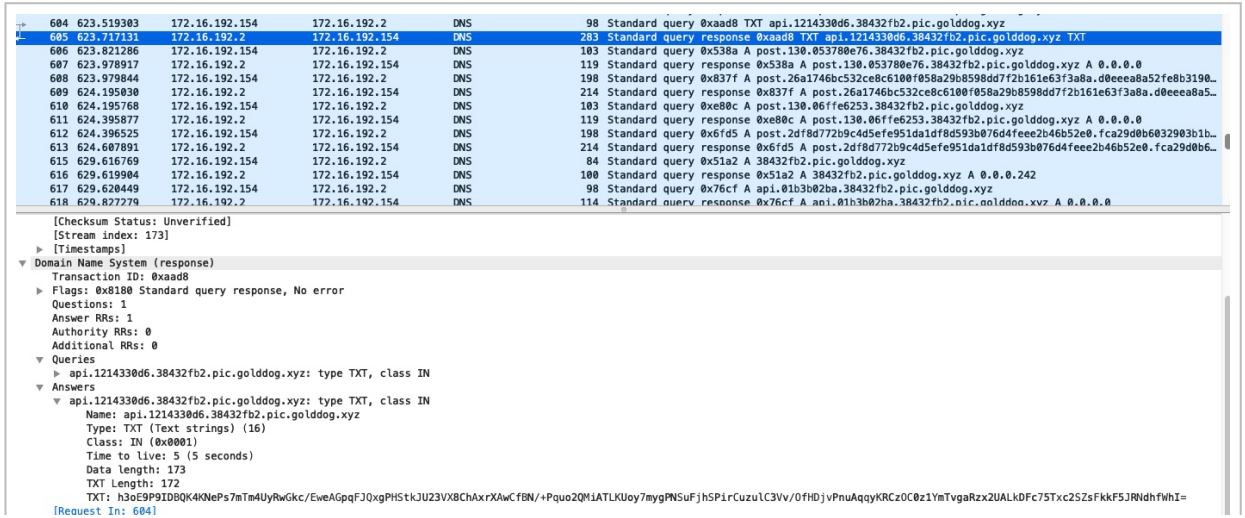


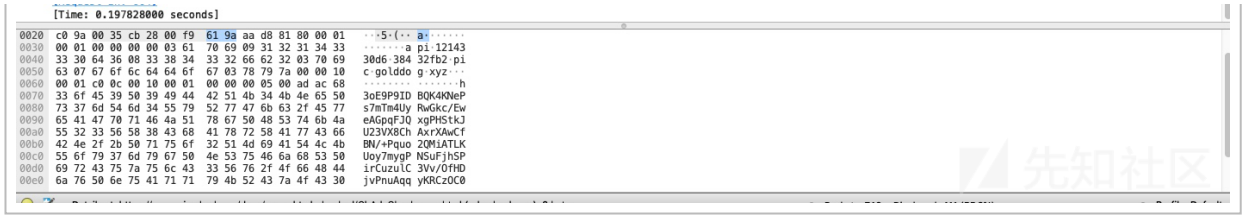
(https://xzfile.aliyuncs.com/media/upload/picture/20200624095236-60ebf338-b5bd-1.jpeg)

随后执行一条命令，可以发现同一时间的流量中夹杂了 TXT 记录：



(https://xzfile.aliyuncs.com/media/upload/picture/20200624095310-74e80840-b5bd-1.jpeg)



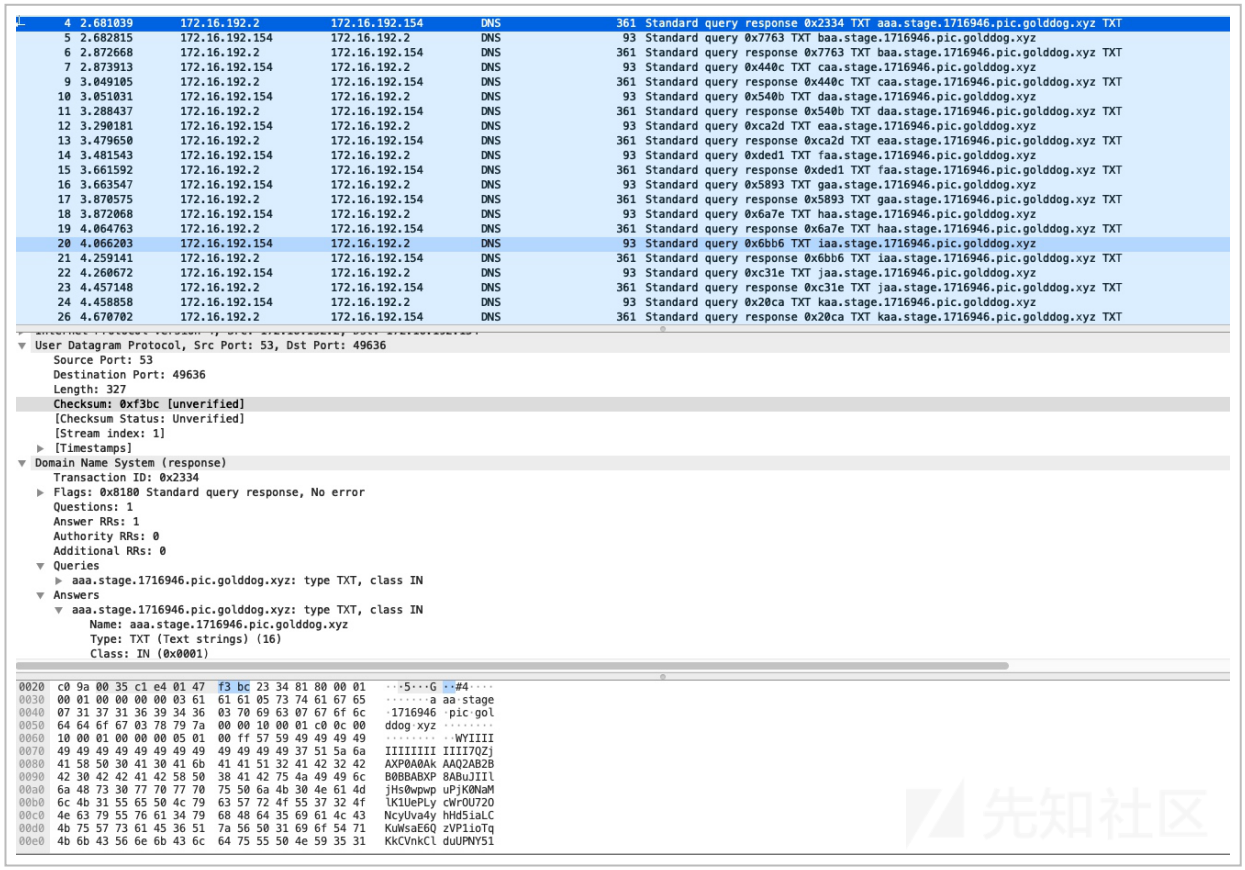


(https://xzfile.aliyuncs.com/media/upload/picture/20200624095325-7ddee91e-b5bd-1.jpeg)

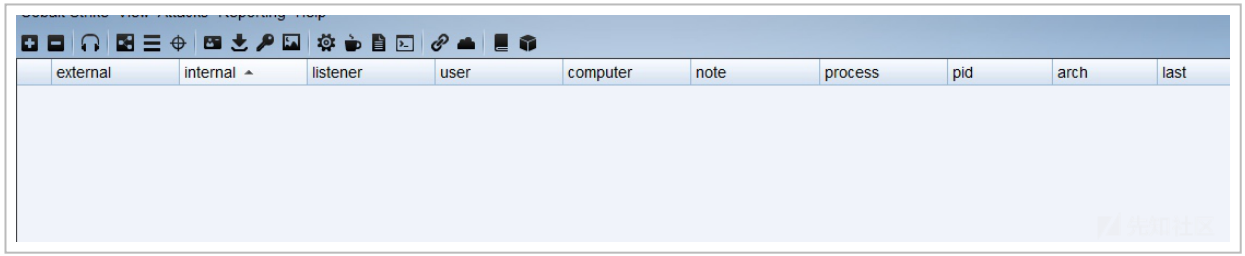
根据官方文档描述，CS4 中有三种数据传输模式，A、AAAA、TXT，默认是 TXT，因此这里的 A 记录只是维持心跳和基本的通信，传输数据默认还是走 TXT 的。

stager

由于 stager 需要下载 stage 组成马子的真正功能体，因此在一开始就需要有一个下载数据的过程。
执行的瞬间会产生大量的 TXT 请求，其中数据是加密后 Base64 编码的，无从得知其中的内容（想想也知道里面就是 stage）。在这个过程中是没有 Ghost Beacon
上线的：

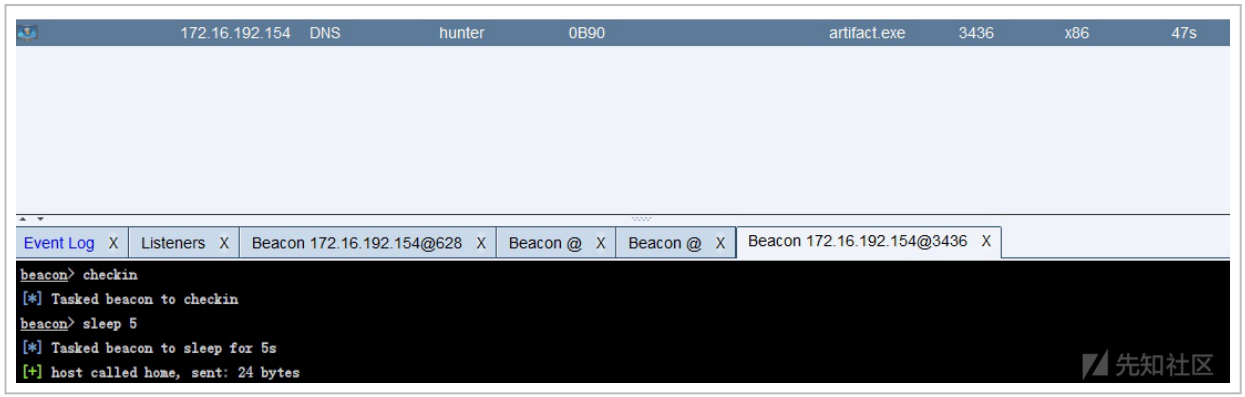


(https://xzfile.aliyuncs.com/media/upload/picture/20200624095347-8ad4cc2e-b5bd-1.jpeg)



(https://xzfile.aliyuncs.com/media/upload/picture/20200624095412-99c0d750-b5bd-1.jpeg)

静候 800 多秒，终于上线了：



(https://xzfile.aliyuncs.com/media/upload/picture/20200624095532-c9911f62-b5bd-1.jpeg)

之后的通信就没有什么特别的了，只是稳定性依然取决于网络情况，如果服务器在国外的话依旧不太稳定。

0x06 关于 BUG

经反复测试，目前网上可以下载的那个破解版是有 BUG 的，checkin 命令无效，teamserver 服务端并不会响应传回启动指令。
但根据 CS 官方手册的描述，执行命令的时候会自动进行 checkin，经测试在 Ghost beacon 出现后输入 mode dns，就可以成功 checkin。后续如果嫌 A 记录通信缓慢，可以再用 mode dns-txt 切换回来。

0x07 关于隐蔽通信

DNS 隧道除应对 TCP 不出网的场景外，其本身的隐蔽通信能力也比较哈好，推荐两个基于 DNS 做隐蔽通信的工具：
chashell (https://github.com/sysdream/chashell)
DNSlivery (https://github.com/no0be/DNSlivery)
此外还有个 ICMP 隧道的工具，思路也可以借鉴：
icmptunnel (https://github.com/DhavalKapil/icmptunnel)
但这款工具并没有对封装的流量做任何加密处理，如果要应用在实战中还需要对代码做一些修改。