# Solving the 7777 Botnet enigma: A cybersecurity quest

By Sekoia TDR, Felix Aimé, Pierre-Antoine D., Charles M., Grégoire Clermont and Jeremy Scion

Published: 2024-07-23 · Archived: 2026-04-06 00:18:49 UTC

## Key Takeaways

- Sekoia.io investigated the mysterious 7777 botnet (aka. Quad7 botnet), published by the independent researcher Gi7w0rm inside the "The curious case of the 7777 botnet" blogpost.
- This investigation allowed us to intercept network communications and malware deployed on a TP-Link router compromised by the Quad7 botnet in France.
- To our understanding, the Quad7 botnet operators leverage compromised TP-Link routers to relay password spraying attacks against Microsoft 365 accounts without any specific targeting.
- Therefore, we link the Quad7 botnet activity to possible long term business email compromise (BEC) cybercriminal activity rather than an APT threat actor.
- However, certain mysteries remain regarding the exploits used to compromise the routers, the geographical distribution of the botnet and the attribution of this activity cluster to a specific threat actor.
- The insecure architecture of this botnet led us to think that it can be hijacked by other threat actors to install their own implants on the compromised TP-Link routers by using the Quad7 botnet accesses.

## Table of contents

## Introduction

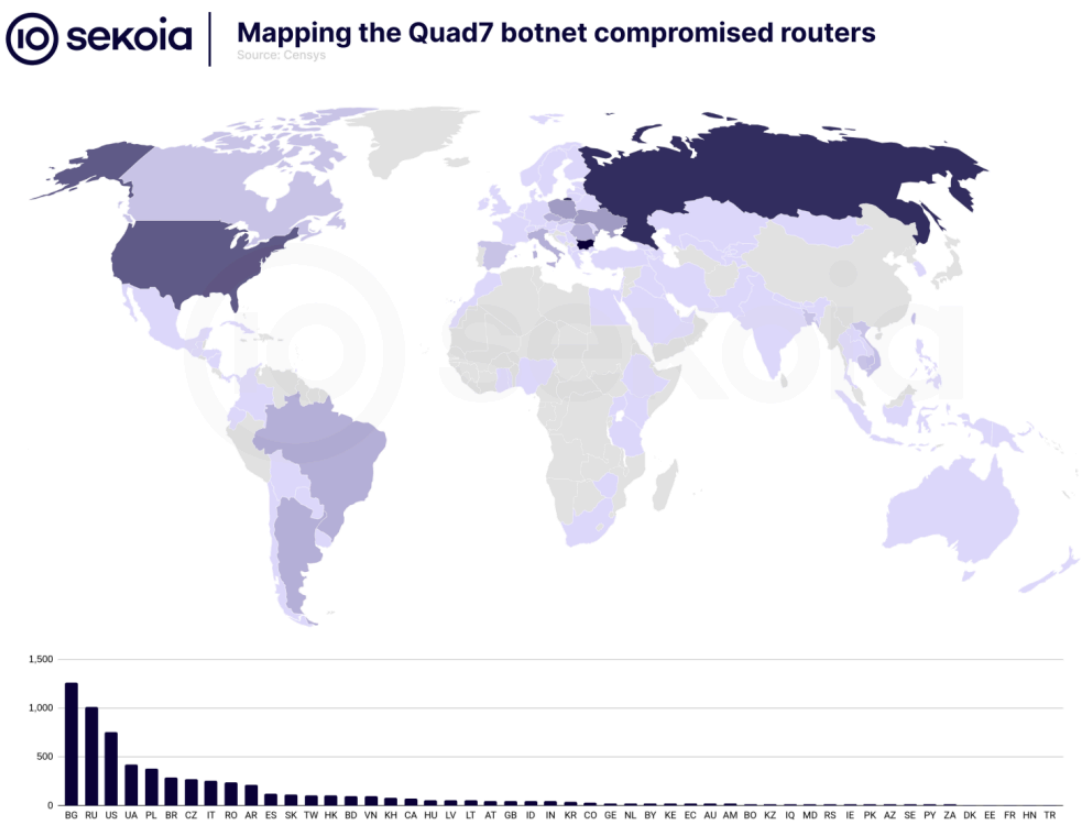
On October 19, 2023, independent researchers Gi7w0rm and Dunstable Toblerone published a blog post about **a botnet nicknamed the `Quad7` or `7777` botnet**, related to the TCP port 7777 opened on compromised devices displaying a mysterious xlogin: banner. This botnet is known in open source for deploying **Socks5 proxies on compromised devices to relay extremely slow "bruteforce" attacks against Microsoft 365 accounts** of many entities around the world.

At Sekoia.io, we have detected these attacks on 0.11% of our monitored Microsoft 365 accounts and have been tracking this botnet since our Intrinsec colleagues shared their findings with us. As this botnet was quite mysterious, targeting our customers and nobody had published on it since Gi7w0rm's blog post, "The Curious Case of the 7777 Botnet," we decided to investigate it.

This blog post will present the full investigation, our successes, and our failures, as it is always interesting to be transparent and provide feedback to the threat intelligence community and teams that may deal with similar IOT/SOHO threats in the future.

## Are all of these compromised TP-Links?

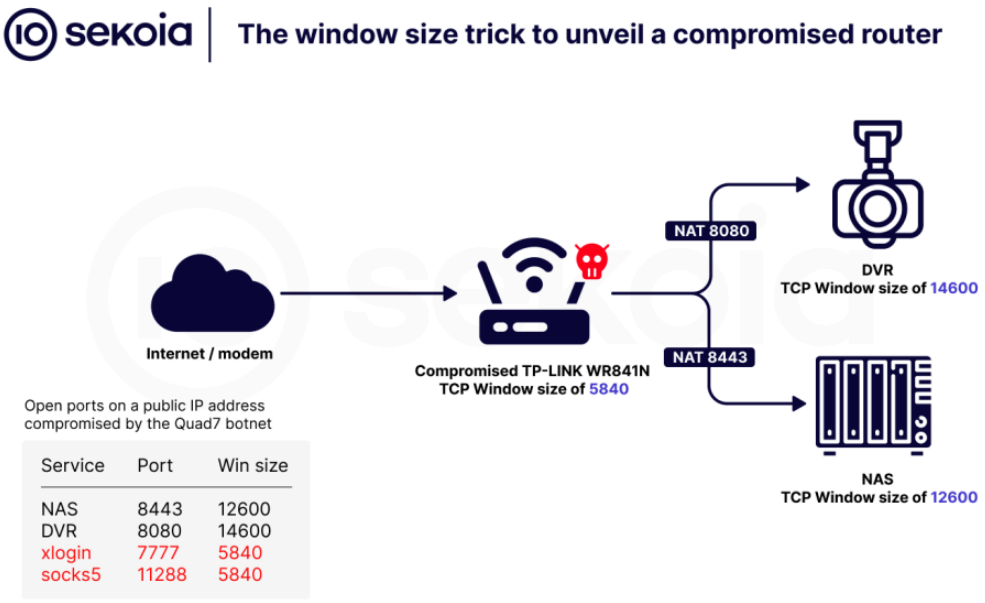
When we started our investigation on this threat, we began by examining what kind of assets had been compromised. This botnet is quite old and constantly evolving, with **the number of unique IP addresses involved dropping from 16,000 in August 2022, to ~7,000 in July 2024**. The geographic distribution of compromised devices is quite surprising, as Bulgaria remains the most infected country, followed by Russia, the US, and Ukraine, as shown below.




According to open-source publications, the Quad7 botnet is suspected to target different kinds of IOTs including IP cameras or NAS devices and SOHO routers, predominantly TP-Link. **However, our investigation found that almost all – we cannot be completely certain – compromised assets were in fact TP-Link routers**.

The bias in the analysis of compromised assets results from the fact that the **operators of the Quad7 botnet try to disable the TP-Link management interface after compromising it** by stopping the binary acting as a web server. Therefore, no TP-Link associated interface or banner is present in many results of online scanners such as Shodan or Censys.

To confirm this hypothesis, we identified valuable information on the TCP windows size returned by compromised assets on the TCP ports 11288 and 7777. We used hping3 tool to scan compromised IP addresses and **it turns out that most of the compromised devices participating in the Quad7 botnet have a windows size known to be related to old versions of the Linux kernel used by TP-Link routers.** For many TP-Link products two TCP windows size values stand out: 5840 (mostly) and 5760.

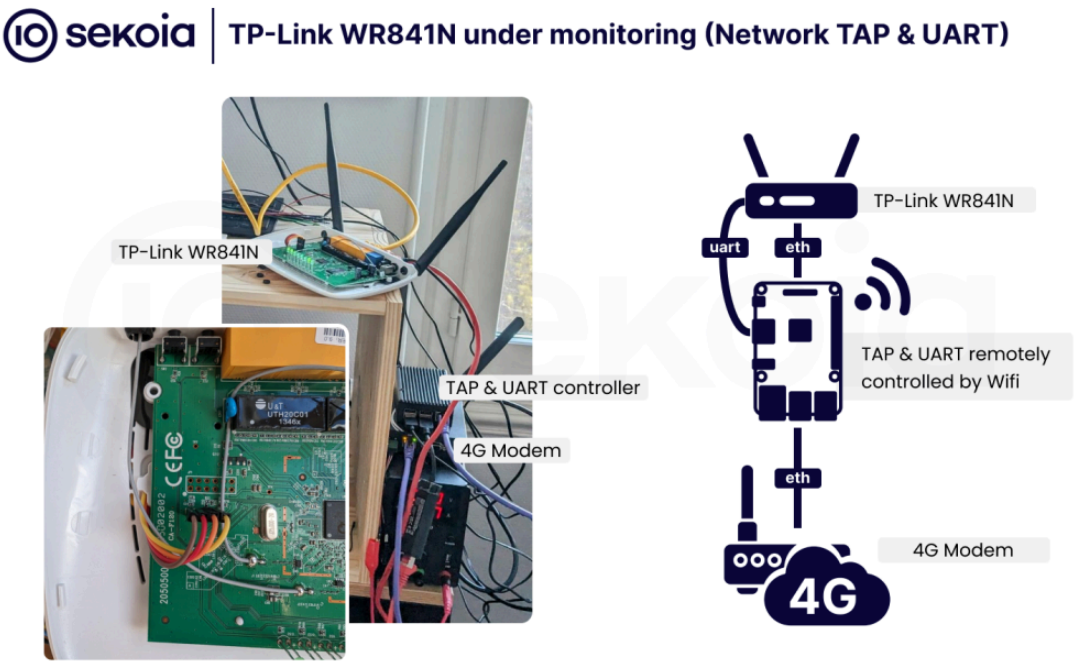



We manually checked some other strange window sizes that we had on approximately fifty IP addresses and according to online scanners history their TP-Link administration panels were open on the Internet in the past. These strange window sizes can sometimes be explained, for example sometimes satellite-link providers are expanding window size for performance gain. Therefore, at this time we were convinced that **the Quad7 botnet operators had at least one exploit chain to gain a remote code execution (RCE) against several management interfaces of TP-Link products**.

## First attempts to catch the Quad7 botnet

Based on these initial findings, we decided to monitor a TP-Link WR841N (firmware: 3.16.9 Build 150320 Rel.57500n) router for a few months. This model is the most compromised according to Censys, with a firmware version known to be vulnerable to the Quad7 botnet. We provided access to the router from **five different IP addresses** (three residential IPs in France, one mobile IP in the UK, and one VPS in Bulgaria, the most impacted country).

The router was fully monitored, including its processes, file system, and network activity. **We created a setup to conduct remote live forensic analysis whenever something suspicious caught our attention**. To do so a Raspberry Pi was connected to the router via UART, serving also as a network tap on the WAN interface, as illustrated in the diagram below. The UART access enabled us to receive alerts via our internal instant messaging application for any suspicious activity in the /tmp/ directory – as the rest of the filesystem is read-only – and at the running processes level.




On the other hand, the network tap wasn't merely a simple network bridge running tcpdump. We utilised the well-known Python Scapy library to monitor and alert us – via our internal instant messaging service too – about any authenticated access to the management interface and the exploitation of standard vulnerabilities such as command injection, file disclosure, etc. The aim was to identify the vulnerabilities exploited by the Quad7 operators.

As we were unaware of the exact exploit chain used by the Quad7 operators to achieve remote code execution, **we also employed Scapy to dynamically modify authentication attempts. This enabled us to accept any credentials provided by attackers** attempting to access the management interface, thereby allowing us to observe the final RCE exploitation, if any.

**Capturing IOT/SOHO threats with honeypots?**

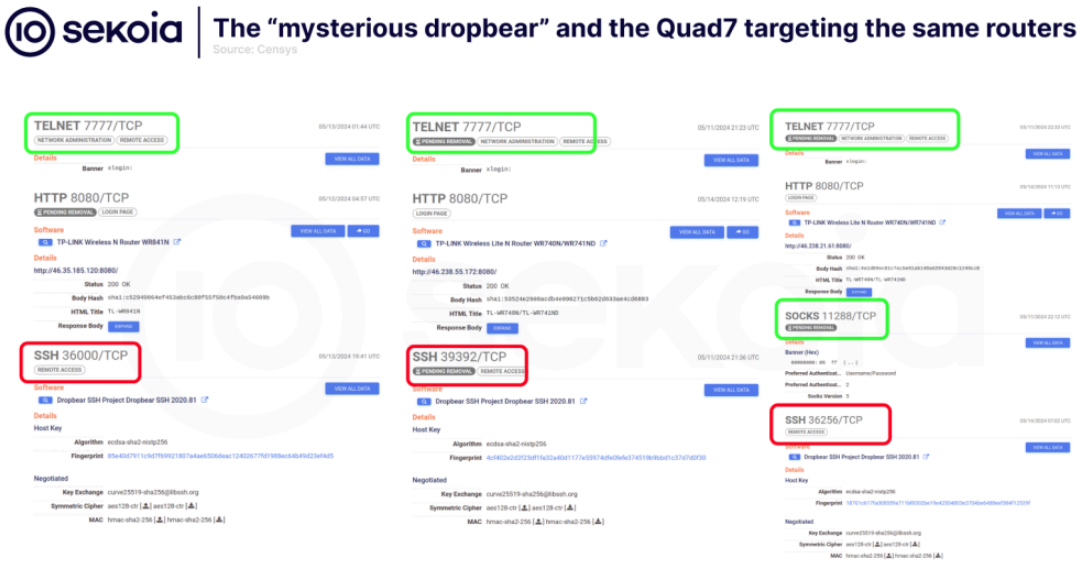
When we set up this system, we were quite enthusiastic about seeing some attacks. However, regarding the Quad7 botnet, we were less enthusiastic as we knew that it seemed to be using an outdated list of IP addresses as targets. Therefore, deploying honeypots with IP addresses that were not on the threat actor list of targets would not allow us to be attacked.

Honeypots are effective tools against standard threats, such as the general noise of cybercriminal activity on the internet (brute force attacks, scanning, and remote code execution at scale when a new CVE is published). However, capturing something more specific is much more difficult, as some threat actors target only residential IPs, specific ASNs or conduct reconnaissance before deploying their final payload to ensure that the targeted device is genuine.

We waited less than a week before observing a notable attack that chained an unauthenticated file disclosure which seems to be not public at this time (according to a Google search) and a command injection. This

unauthenticated file disclosure allowed the attacker to retrieve the pair of credentials stored in /tmp/dropbear/dropbearpwd, to replay them in the HTTP Basic authentication of the management interface. Once authenticated, the attacker exploited a known command injection vulnerability in the Parental Control page to achieve the RCE.

We still don't know the goal of this attack, as the attacker launched Dropbear (a pre-installed lightweight SSH agent) on a higher port, transferred his own BusyBox via the created SSH session, and then left the router after cleaning up their traces. However, it is interesting to note that this threat actor also targeted IP addresses compromised by the Quad7 botnet.




**Despite finding an overlap on more than 80 IP addresses between the two attacks during our investigation, we do not believe they are related.** This threat actor engages in compromised SOHO hopping (the attack originated from a compromised D-Link router) and utilises SSH for file transfer unlike the Quad7 operators who use the TFTP protocol. Furthermore, this actor does not deactivate the management interface of the compromised router after exploiting it. **Consequently, we occasionally observe this actor compromising routers prior the Quad7 botnet operators, who then most of the time close the management interface.**

We also observed during this monitoring brute force attempts of the HTTP Basic authentication, exploitation of known file disclosure vulnerabilities affecting TP-Link devices, and instances of DNS records being altered to redirect users to rogue DNS servers for ad distribution. However, these activities seemed more related to **standard noise of SOHO/IOT targeting** than the Quad7 operations.

**It remains unclear why the Quad7 operators persist in maintaining the infrastructure established in 2022 by re-compromising routers upon their restart**, rather than expanding their botnet by targeting new IP addresses. One possible reason could be to evade detection by honeypots, as new IP addresses after the `The curious case of the 7777 botnet` may be honeypots to catch them. **Another, and more plausible, explanation is that they haven't updated their target list for months or even years**. This hypothesis would also explain the decrease of compromised assets over the time.

## Identifying victims

As our honeypot didn't yield the expected results, we shifted to a different strategy: identifying some victims in France. Of the eleven IPs observed in 2024, **we were able to identify and contact three individuals, requesting their assistance** in tapping their routers and physically recovering the Quad7 botnet related malwares.

Why intervene physically when a victim could simply send us their router? The reason is simple: the majority of the file system is read-only (squashfs), and the /tmp/ directory is writable – but in volatile memory. **As soon as the router is unplugged, its file system would be reset, making it impossible to retrieve the malicious codes.**

**Fortunately, one of these attempts was successful, providing us with more insights into the Quad7 botnet operations.** Of the other two, one individual replaced his router after receiving our email but did not reply favourably, and the third did not reply at all, possibly thinking our message was a scam.

## Physical intervention preparation

The compromised router was an old Archer C7 v2.0 (Firmware: 3.15.3 Build 180305 Rel.51282n). Our contact promptly understood the situation, our intentions regarding his router and graciously allowed us access to it. We agreed to plan an operation comprising two main tasks:

- **Sending a 'plug and play' in-line network tap** to install prior to our intervention. This tap monitored only ports TELNET/7777, SOCKS/11288, and the management interface port if the router was restarted. The goal was to understand the upper level of the infrastructure related to the Quad7 botnet and its bruteforce activities.
- **Physically access the router via UART to get a root shell on it**. The aim of that was to retrieve the malware related to the Quad7 botnet. At this time, we were uncertain if the infamous `xlogin:` service listening on the TELNET/7777 port was a kind of bind shell access or was merely a decoy from the attackers.

This intervention involved two potential points of failure to address.

**The first issue was technical problems with the in-line network tap**, such as bandwidth throttling or running out of disk space due to the captured data. To address this point of failure, we created a cheap tap with two Ethernet ports of 1Gbps each. The setup was straightforward: the two interfaces were bridged, and a tcpdump was initiated by a cron job at reboot to save network captures in the home directory, as illustrated below.

@reboot /bin/bash -c 'sleep 60 && UUID=$(uuidgen) && /usr/bin/tcpdump -i br0 "port [admin port] or port 7777 or port 11288" -w /home/tap/capture_${UUID}.pcap -C 1000 -W 10' >> /home/tap/tcpdump.log 2>&1

To prevent the SD card in our tap from running out of space, and since we were uncertain about the network flows passing through the socks proxy implemented by the Quad7 operators, we decided to create a rotation of the pcaps with a maximum of 10Gb of captured network flow for each reboot of the tap. Our contact simply needed to connect the tap, and without requiring any command line or additional instructions, the tap immediately began capturing network data.

**The second technical issue was related to the UART itself**. From an attacker's perspective, It is straightforward to disable the user authentication from UART after having compromised the router simply by deleting or rewriting one specific file inside the /tmp/ directory. Consequently, if we cannot log in, we cannot get the malicious codes.

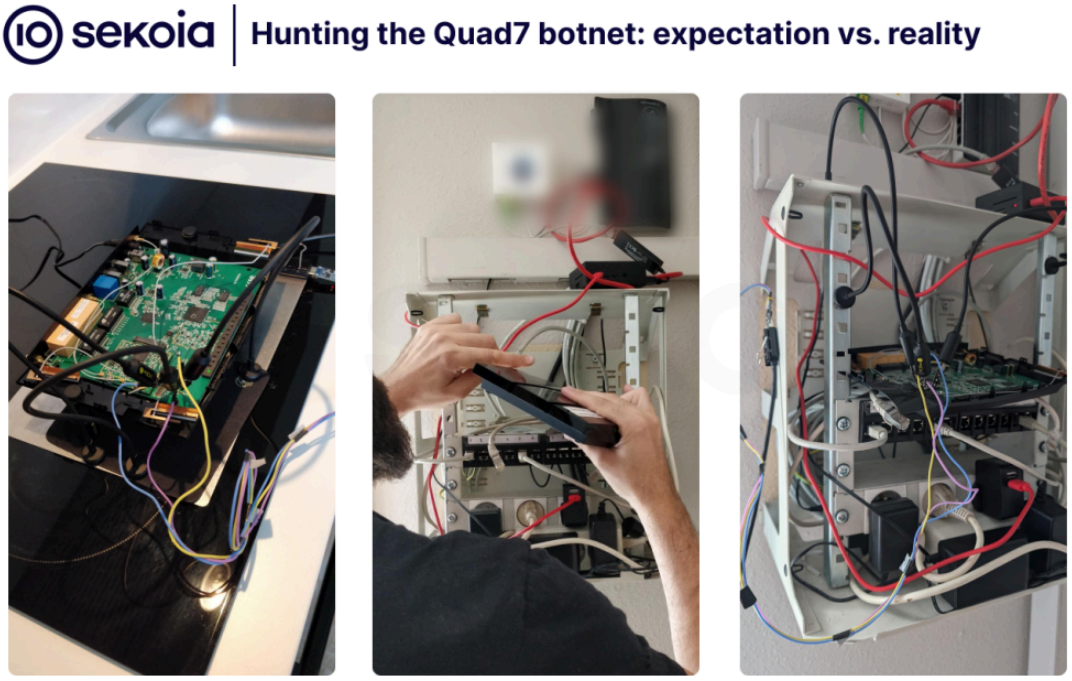
To address that issue, we decided to have a backup plan prior to our intervention. **We developed our own remote forensics-friendly reverse shell specifically for the Archer C7**. The idea behind this was simple: since we knew that the threat actor re-compromises the router following a reboot, **we could pre-implant it to get his malware**. To accomplish this, we just have to reboot the router and pre-implant it with our own reverse shell via the UART access (re-enabled following the router's reboot), and then wait for the attacker to drop his codes.

After conducting numerous tests in real conditions, including opening the router without unplugging it, and preventing connection loss by manipulating it, which could lead to a router reboot, we were prepared to travel 1,000 kilometers across France to access this router.

## The intervention: when expectation meets reality

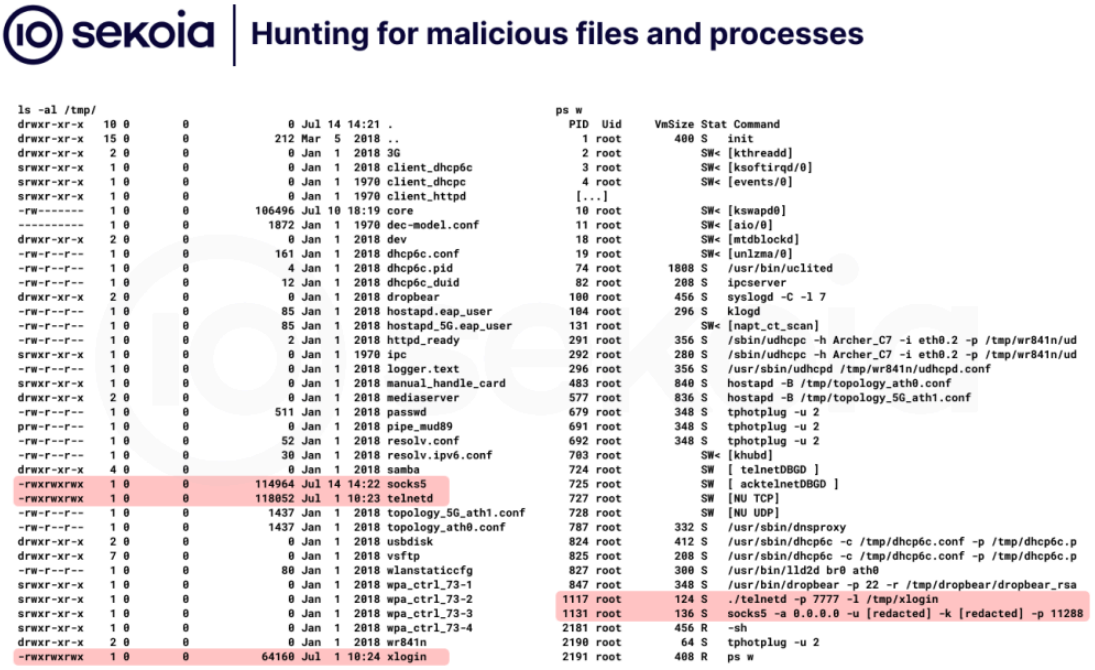
For this intervention, our team consisted of three individuals: a reverser to analyse the obtained malware, a support member to provide valuable inputs/ideas and one person to handle the router. **The critical aspect was to open and connect to the UART without unplugging the router or causing connection loss, as this could result in the loss of the malware.** We extensively tested this configuration using a PCBite, even on hot plates with a similar width to a server cabinet (24 inches) as our contact informed us that the router was located in "a small server cabinet".

Upon arriving at the office where the router was hosted, we faced our first challenge: we encountered a very small switch cabinet with limited space to manoeuvre the router. Consequently the operation to remove the Phillips screws, open the router and connect probes to the UART was quite intricate. However, thanks to PCBite, we were able to magnetise the probes to the cabinet itself during the operation, ensuring stability of the UART connection.




Fortunately, the attacker had not disabled the user authentication. Consequently, we obtained a root shell within seconds, enabling us to examine suspicious processes and network connections. **Prior to the intervention we took several snapshots of the list of running processes and writable directories of a clean router**. Hence a simple diffing between the compromised router and clean snapshots revealed the difference immediately. The attacker did not try to conceal itself by modifying files or process names.

In the process listing below, **you can clearly see the suspicious execution of three binaries: telnetd, xlogin, and socks5**. It is worth mentioning that, as stated before, the attackers deactivated the web interface by killing /usr/bin/httpd, as it is no longer present in the running processes list.



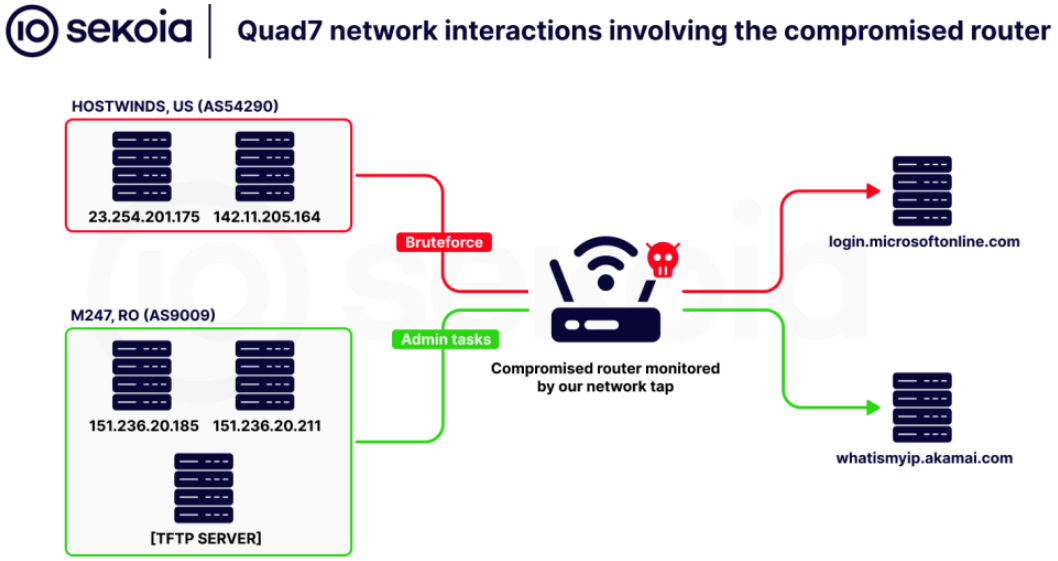

To retrieve the malicious codes from the Archer C7 2.0 using BusyBox, there are two primary methods: using TFTP (which is relatively slow, and does not verify file integrity during transfer) or simply connecting a flash drive formatted in FAT. As we wanted to ensure the integrity of the transferred files and as the Busybox installed on the Archer C7 lacked a reliable utility for this purpose, we uploaded our implant onto the router. This approach provided us with a convenient method to download and verify the integrity of the downloaded files directly.


We successfully obtained the binaries that our attacker pushed onto the compromised router. These binaries consisted of a Telnet binary coming from BusyBox (386bf8259668c0abb6c72fdcae904164) which is listening on the TCP/7777 and redirected incoming connections to a simple authenticated shell named xlogin (69ced04a2ec895084d3aab1086216d32). Additionally, there was a Socks5 proxy (29e6df5bb30ed8fd12c09d9b6890ab4f) derived from the bhhbazinga's Sock5 open source project which listened on the SOCKS/11288 used to relay brute force attacks to Microsoft 365 API endpoints. Despite our expectations, there was nothing else – no sophisticated goodies for our reverser after all.

However, **binaries alone do not provide much insight without some network context around them**. In the next section we present our investigation based on the network capture collected by our network tap to determine if we found valuable data inside.

## Digging into the network capture.

Fortunately, our plug-and-play tap functioned effectively, and we collected one week's data. Sadly, the router did not reboot during this period and was replaced shortly after our operation. As a result, we did not capture any exploits used by the operators to re-compromise it. Nevertheless, we found answers to some of our questions by analysing our small network capture (11MB) obtained from connections to the ports TELNET/7777 and SOCKS/11288. Here is a summary of what we have seen:



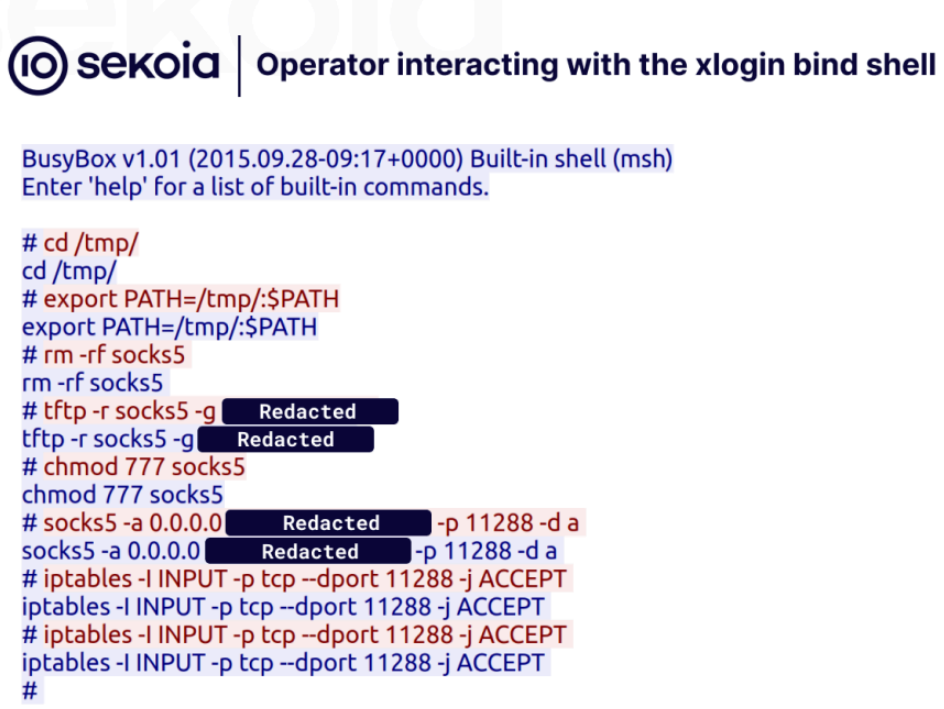

Our first question was whether this botnet is used solely by the Quad7 operators to brute force Microsoft 365 user accounts or other services. **Our network capture revealed that 912 connections were made using the socks proxy during one week to login.microsoftonline.com**. However, there were only a few password attempts (maybe only one?) per connection, which is surprising for a 'brute force attack.' Therefore, **we believe this botnet is primarily used to conduct simple password spraying attacks, rather than engaging in traditional brute force attacks**.

We identified two servers (142.11.205[.]164 and 23.254.201[.]175) both hosted under HOSTWINDS, US (AS54290), that authenticated themselves to the socks proxy to send requests to login.microsoftonline.com.

To check the proxy status, two IP addresses were employed (151.236.20[.]185 and 151.236.20[.]211), this time hosted under M247, RO (AS9009). These IP addresses used the socks proxy to only check the exit node IP address by issuing HTTP requests to whatismyip.akamai.com with Go and Python user agents.

**This initial analysis led us to believe that only one threat actor was using the socks proxy installed on the router to check Microsoft 365 user accounts**. This finding was surprising, given that socks remain an unsecure proxying protocol with an authentication in clear-text.

Our second question was related to the xlogin bind shell and who was interacting with it. We observed the IP address 151.236.20[.]185 doing the same continuous checks to verify that the SOCKS/11288 and TELNET/7777 ports were available. And at one time, **we saw the same IP address issuing hands-on keyboard commands via the xlogin bind shell** to update the socks5 binary on the compromised router.




## Let’s take a look at the brute force attempts in our telemetry.

The rate of attempts per account is on average about one every 40 hours, and it is common for accounts to have received more than **50 authentication attempts over several months**. This suggests that a list of passwords is being tested on each account (password spraying), rather than credential pairs obtained from data breaches (credential stuffing).

In the dataset of Sekoia.io XDR customers, **27% of organisations (Entra ID tenants) were targeted over the last 30 days**. On average, 0.11% of monitored user accounts are affected, with some larger tenants having several dozens of targeted accounts, and some smaller organisations having up to 5% of their users targeted.

Several industry verticals, including local authorities, have been targeted in recent days **without any discernible pattern relating to the type of industry or the size of the targeted entity**.

Regarding the targeted accounts, despite what is said in the original blogpost, no clear pattern emerges from the targeted accounts. Indeed, **while there are some VIPs, they are very much in the minority compared to the bulk of the tested accounts**. Some are not even real user accounts but rather aliases such as unsubscribe@, noreply@, or even customersupport@. The list of targeted email addresses evolves over time, with new addresses being added to or removed from the list.

Therefore, **the threat actor behind the Quad7 botnet does not appear to be an APT but more likely cybercriminals looking to carry out business email compromise via password spraying from compromised SOHO routers**.

## After this investigation, some mysteries remain.

**The primary mystery of this investigation remains the attribution question**. Many speculations have been made in open-source discussions, suggesting it could be another Chinese IoT botnet or a North Korean campaign. However, during our investigation, **we were unable to identify any evidence pointing to a known threat actor**. Their binaries are stripped and only one is really custom, the xlogin shell. **The time frame of the brute force attacks did not reveal any significant information either**, as the attempts were inactive on some days, and we monitored this router for one week only.

Regarding the infrastructure, we found only one interesting match with a C2 related to GobRAT – another notable IoT botnet first disclosed by the JP-CERT – next to one of the Quad7 botnet operators' IP addresses. However, this is a very weak link as GobRAT and the other implants dropped by the same threat actor are modular and way more sophisticated than a simple bind shell with a socks proxy.

**The second mystery is the exploited vulnerabilities used to compromise these routers.** We are almost certain that the attackers had an authentication bypass leading to an RCE or an unauthenticated RCE. As TP-Link reused almost the same firmware codebase between some router series – you can see that with WR841N related files on the Archer C7 directory listing screenshot – it is likely possible that the attackers are using only one exploit chain that affects all routers.

**The last remaining mystery is the geographical distribution of this botnet.** Initially, we thought that it might be due to default passwords set by internet providers in specific countries. However, we are confident that the operators behind the Quad7 botnet are not doing brute-forcing attacks against the router's management interface. Therefore, this mystery remains unsolved.

In summary, the Quad7 botnet still has many unanswered questions, and we believe that companies with a better view on the infrastructure supporting this botnet will be able to provide answers. It is important to remember that our investigation was limited to one router and one week of monitoring.

### Threats targeting edge devices: an unseen but prevalent danger

The Quad7 botnet operators made various mistakes, allowing the community to identify the compromised routers through internet scanners. However, the vast majority of threats targeting edge devices remain invisible. Therefore, we encourage restricting the remote administration of these devices to specific IP addresses to prevent the exploitation of vulnerabilities and ensure your devices are updated with the latest firmware.

In the case of enterprise-grade edge devices (e.g., VPN/mail gateways, routers, etc.), many threat actors are using them to gain a foothold inside networks while staying under the radar of security solutions. Consequently, we recommend deploying specific rules to monitor authentication attempts against internal servers originating from these edge devices' IP addresses and other unmonitored devices inside your network.

## Detection Bonus: Advice for MSSP to hunt for Quad7 O365 targeting.

The authentication attempts performed via the Quad7 botnet botnet feature a fairly unique combination of attributes that make them easily identifiable in the Entra ID Sign-in log or Microsoft 365 audit log:

User-Agent is either:

- Mozilla/5.0 (Windows NT 10.0; WOW64; Trident/7.0; rv:11.0) like Gecko
- Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.149 Safari/537.36

Application ID: 1950a258-227b-4e31-a9cf-717495945fc2 (Microsoft Azure PowerShell)

Resource ID is either:

- 00000003-0000-0000-c000-000000000000 (Microsoft Graph)
- <empty>

It is possible to confirm that the request originated from the botnet by looking up the source IP address in a tool like Censys or Shodan to confirm the presence of the “xlogin:” banner on port 7777. In the dataset of logs available to Sekoia.io, over 98% of the authentication attempts identified by this heuristic can be conclusively attributed to the Quad7 botnet.

This query should return almost exclusively authentication failures, with the most common error codes being:

- 50126: Invalid username or password
- 50053: Account locked
- 50057: Account disabled

Authentication attempts resulting in other status codes should be investigated to determine if the account’s password was discovered. When the logs indicate that the authentication was blocked by multi-factor authentication or conditional access policies, the password should be considered compromised, as these verifications only occur after a correct password was submitted. Identifying these authentication logs can help assess how an organisation is targeted.

**Thank you for reading this blog post. Please don’t hesitate to provide your feedback on our publications by clicking here. You can also contact us at tdr[at]sekoia.io for further discussions.**

## Quad7 Botnet Indicators of compromise

*Some IOCs & samples aren’t shared publicly. If you are a national CERT or LEA, we can share the IOCs and the samples with you under TLP:AMBER+STRICT as they contain hardcoded passwords leading to remote shell execution on the compromised routers.*
*Please contact tdr [ at ] sekoia [ dot ] io*

**Infrastructure**

```
142.11.205[.]164
23.254.201[.]175
151.236.20[.]185
151.236.20[.]211
[TFTP SERVER: on demand]
```

### Malware

```
98d3764862b182417c910a96e0fbfe71  init.sh
c8e229bed1659f1613c1016b3345ef08  microsocks
29e6df5bb30ed8fd12c09d9b6890ab4f  socks5
69ced04a2ec895084d3aab1086216d32  xlogin
```

### Yara rules

```
rule unknown_7777_xlogin {
    meta:
        id = "01510244-0795-4299-aa66-056a2b4682e7"
        version = "1.0"
        intrusion_set = "Quad7 Botnet"
        malware = "xlogin"
        description = "Detects the xlogin bind shell"
        source = "Sekoia.io"
        creation_date = "2024-07-18"
        classification = "TLP:CLEAR"
        hash = "69ced04a2ec895084d3aab1086216d32"
    strings:
        $ = { 2f 62 69 6e 2f 73 68 00 2f 74 6d 70 2f 6c 6f 67 69 6e }
        $ = { 2F 64 65 76 2F 6E 75 6C 6C 00 00 00 2F 62 69 6E 2F 73 68 00 2D 63  }
    condition:
        uint32be(0) == 0x7f454c46 and
        filesize &amp;lt; 100KB and
        all of them
}

rule tool_bhhbazinga_Sock5_strings {
    meta:
        id = "45dafe1e-81bb-4202-93ab-fcd46e8d8d6b"
        version = "1.0"
        tool = "bhhbazinga sock5"
        description = "Detects the sock5 project developed by bhhbazinga"
        source = "Sekoia.io"
        creation_date = "2024-07-18"
        classification = "TLP:CLEAR"
        hash = "29e6df5bb30ed8fd12c09d9b6890ab4f"
```

```
    strings:
        $ = "tunnel_read_handle"
        $ = "tunnel_write_handle"
        $ = "tunnel_open_handle"
        $ = "tunnel_auth_handle"
        $ = "tunnel_connect_to_remote"
        $ = "tunnel_request_handle"
        $ = "-u<optional> : username"
        $ = "-k<optional> : password"
        $ = "d<optional> : backgroud"
    condition:
        uint32be(0) == 0x7f454c46 and
        filesize &lt; 150KB and
        5 of them
}

rule tool_microsocks_strings {
    meta:
        id = "07cdecee-3a62-4e1d-96cd-24e4dc30925d"
        version = "1.0"
        tool = "microsocks"
        description = "Detects the microsocks project developed by rofl0r"
        source = "Sekoia.io"
        creation_date = "2024-07-18"
        classification = "TLP:CLEAR"
        hash = "c8e229bed1659f1613c1016b3345ef08"
    strings:
        $ = "error: user and pass must be used together"
        $ = ": option requires an argument:"
        $ = "option -1 activates auth_once mode:"
        $ = "error: option -%c requires an operan"
        $ = "client[%d] %s: connected to %s:%d"
    condition:
        uint32be(0) == 0x7f454c46 and
        filesize &lt; 100KB and
        4 of them
}
```

### Sigma rule

```
title: Entra ID Account Password Compromised By 7777 Botnet
description: Detects a successful Entra ID authentication featuring characteristics associated with
author: Sekoia.io
date: 2024/07/19
tags:
    - tlp.clear
```

```
logsource:
    product: azure
    service: signinlogs
detection:
    selection:
        userAgent:
            - 'Mozilla/5.0 (Windows NT 10.0; WOW64; Trident/7.0; rv:11.0) like Gecko'
            - 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chro
        AppId: 1950a258-227b-4e31-a9cf-717495945fc2 # Microsoft Azure PowerShell
        ResourceId: 00000003-0000-0000-c000-000000000000 # Microsoft Graph
    filter:
        ResultType:
            - 50126 # InvalidUserNameOrPassword
            - 50053 # IdsLocked
            - 50057 # UserDisabled
            - 50056 # InvalidPasswordNullPassword
    condition: selection and not filter
falsepositives:
    - Other error codes indicating that the password was incorrect.
level: high
```

Notes:

- A similar rule can be written for the Microsoft 365 audit log.
- For simplicity, this rule omits the case where ResourceId is empty, as this variant has not been observed recently.

## Greetings

We would like to thank Intrinsec for putting us on the trail of this botnet and Gi7w0rm for its publication. Thanks to Nicolas Noël (Disweb.fr), who allowed us to intervene on one of his client’s routers, as well as to everyone who, directly or indirectly, contributed to the successful completion of this operation.

Share

7777 botnet Botnet CTI Malware Quad7 botnet

Share this post:

---