QIDI Security Concerns

29th May 2024

Introduction

This document serves to record my remarks regarding the security posture of the 3d printer manufacturer Ruian Qidi Technology henceforth referred to as Qidi. My intention is to highlight potential vulnerabilities and articulate their impact to end users without being defamatory. I will aim to remain impartial and factual throughout.

In the interest of responsible disclosure, I intend to permit Qidi 90 days following notification to remediate all findings before disclosing this document publicly.

UPDATE 2024-10-20

It has now been several months since authoring this report and having worked with Qidi through the responsible disclosure process. I have reviewed the remedial efforts and updated this report to reflect the current state.

My experience with Qidi has been mixed. They responded positively to my initial report and to their credit, have successfully remediated some findings. Unfortunately, the most severe findings were not effectively addressed and remain critical.

I would also like to thank Angus from Maker's Muse for connecting me with representatives at Qidi and Chloe from Qidi for mediating communications between myself and the engineering teams.

Scope

My case study has focused on the recent Qidi Q1 Pro model and its cloud connected offerings although the applicability of findings likely extend to other models and services. The latest version at the initial time of writing was `V4.4.19`.

Constraints

The scope of this analysis is vast including printer firmware, mobile applications, git repos and cloud infrastructure. Investigating and documenting this takes a lot of personal time and energy so I have endeavoured to strike a balance between due diligence and resources. I do not own the printer in question so printer specific finding validation was limited to source code review, however cloud related finding validation was able to be conducted against live servers.

Please frame this document with the awareness that I did **not** find every issue and was unable to validate every finding to the extent I would have desired.

Executive Summary

Multiple significant deficiencies have been identified which may result in total system compromise of vulnerable 3d printers by remote adversaries with low exploit complexity. The overall security posture of Qidi remains poor demonstrating lacking operational and developmental security.

The potential impact of such compromise could result in serious injury or loss of property.

Table with findings by CVSS 4.0 severity:

Severity	**Count before Remediat-ion**	**Count after Remediat-ion**
Critical	2	2
High	3	0
Medium	0	1
Low	0	0
Info	4	2

It appears the culture at Qidi is to prioritise development of new products and services without any regard security. This product-first mentality is common in industry and may be due to lack of expertise, finances or leadership values. I would like to comment that I have found no evidence to substantiate the claim that Qidi have intentionally distributed malware in / backdoored their products.

Findings

1. QIDI Link Lacks Authentication

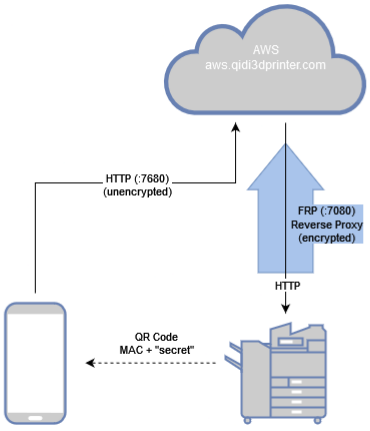
CVSS 4.0
- **Severity**: Critical
- **Score**: 9.3
- **Vector**: CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:H/VI:H/VA:H/SC:N/SI:N/SA:N

Finding
The method of establishing connectivity between the mobile app QIDI Link and the cloud infrastructure lacks any form of authentication.

Connectivity is achieved in the following fashion:
1. Printer generates a FRPC (reverse proxy) config using its wifi dongle MAC address and a random integer between 100,000 and 999,999 as the subdomain. E.g. `001122334455_123456` (the MAC will actually default to `000000000000` if it can't be read)
2. This subdomain is shared out of band with the application through a QR code.
3. Printer establishes reverse tunnel with `aws.qidi3dprinter.com:7080` (or `aliyun.qidi3dprinter.com` - commented out) via FRP. This tunnel is persistent so long as the printer is on and the cloud endpoint is reachable.
4. The application connects to `001122334455_123456.aws.qidi3dprinter.com:7680` over plain text HTTP which is then forwarded through the reverse proxy to the local nginx server on the printer (on port `10088`).

The only object an attacker requires to connect to a printer is its MAC address and the random 6 digit integer. The former is either enumerable over a network, predictable given a brand and model of wifi dongle or simply bruteforceable. The latter is very low entropy and easily bruteforceable in under a day.

This "secret" subdomain can also be sniffed from the connection between the mobile application and cloud endpoint through:
- Plaintext HTTP `host` header
- DNS requests (if not using DNSSEC / DoH)
- Even if TLS were used, the `SNI` field during the initial handshake



A simplified diagram of the network topology

```
serverAddr = "54.68.120.215"
serverPort = 7080

auth.method = "token"
auth.token = "qidi tech"

log.to = "/root/frp/frpc.log"
log.level = "info"
log.maxDays = 3

[[proxies]]
name = "{get_printer_model().replace(' ', '_')}_{mac_address}"
type = "http"
localPort = 10088
transport.bandwidthLimit = "100KB"
transport.useEncryption = true
subdomain = "{mac_address}_{random.randint(100000, 999999)}"
```

Excerpt from `/root/frp/setup_frpc.py`

```
[...]
function(t){var i="";if(1==t.connection){var
n=t.mac_address.split(":").join("").toLowerCase();i="http://".concat(n,"_").co-
ncat(t.device_code,".").concat(t.server).concat(".qidi3dprinter.com:
7680"),e("error","deviceUrl",i," at utils/common.js:178")}else
i="http://".concat(t.local_ip,":10088");
[...]
```

Excerpt from QIDI Link android application

Impact

An attacker could steal or guess a printer's subdomain string and remotely gain access to its fluidd interface unbeknownst to the user. From there they would be able to edit `printer.cfg` and gain arbitrary code execution.

This could result in:

- Loss of life - overriding klipper safeguards and commanding heaters (including the 240V mains heater) to run until something shorts or ignites
- Espionage through the onboard camera
- Theft of wifi credentials
- Theft of user files
- Enrolment into a botnet
- Pivotting to more sensitive devices on the LAN
- Denial of Service

Recommendation

- Immediate suspension of all QIDI Link functionality
- The total redesign of the remote connectivity function should consider:
  - ◇ Use modern, well-tested authentication mechanisms
  - ◇ Use high entropy secrets
  - ◇ Store and distribute said secrets through secure channels
  - ◇ Prefer anonymous identifiers such as UUIDv4 over predictable and trackable information such as MAC / IP address

Update 2024-10-01

UNCHANGED

I have identified the following changes:
• The generation of the frpc config has moved from `/root/frp/setup_frpc.py` to `/root/xindi/build` (an in-house written cpp binary)
• The subdomain value appears to generated server side after the printer is registered by making a POST to `https://api.qidi3dprinter.com/qidi/common/updateDeviceWithSn` This value is bound to the user's cloud account and the local ftpc is updated accordingly. I am unable to comment on the entropy of this value as I gave up reversing the binary to understand what parameters the endpoint expects.
• The local port of the frpc reverse proxy has changed from `10088` to `80`

The changes have made code review significantly harder as most operations are now handled by the xindi binary. The fundamental mechanism of customers accessing their printers through the frp server remains the same with the only "secret" being the subdomain. This value is static, outside of the users control and can be sniffed with relative ease.

In summary, in absence of any real authentication mechanisms, I still consider the above finding valid.

2. QIDI Link uses Insecure Protocols

CVSS 4.0

- **Severity**: Critical
- **Score**: 9.2
- **Vector**: CVSS:4.0/AV:N/AC:L/AT:P/PR:N/UI:N/VC:H/VI:H/VA:H/SC:N/SI:N/SA:N

Finding

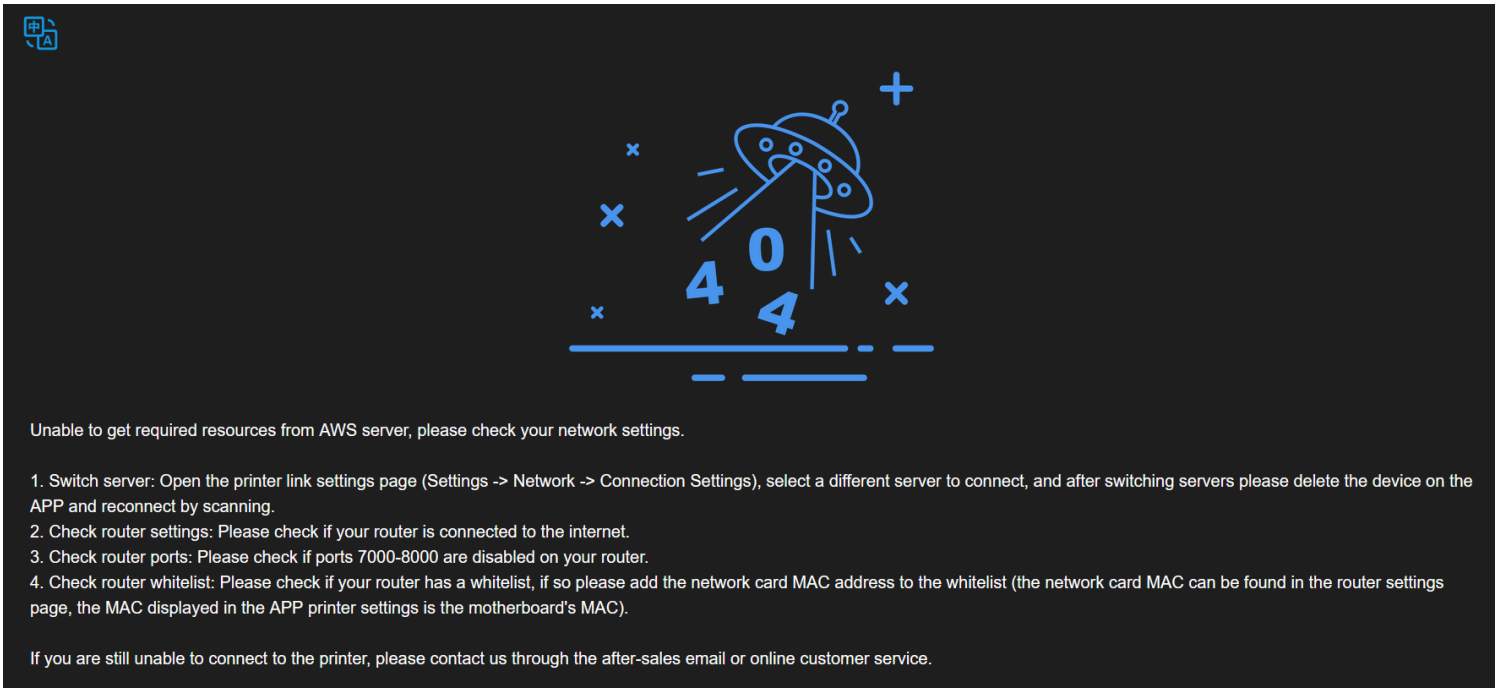
The method of establishing connectivity between the mobile app QIDI Link and the cloud infrastructure uses plaintext HTTP.

An attacker appropriately positioned may intercept and manipulate any packets. For example, an attacker situated on the same public wifi hotspot as the mobile device may suffice.

```
snow@kali ~/qidi $ curl http://000000000000_123456.aws.qidi3dprinter.com:7680 -
v -I
*   Trying 54.68.120.215:7680...
* TCP_NODELAY set
* Connected to 000000000000_123456.aws.qidi3dprinter.com (54.68.120.215) port
7680 (#0)
> HEAD / HTTP/1.1
> Host: 000000000000_123456.aws.qidi3dprinter.com:7680
> User-Agent: curl/7.68.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 404 Not Found
HTTP/1.1 404 Not Found
< Date: Wed, 29 May 2024 16:40:23 GMT
Date: Wed, 29 May 2024 16:40:23 GMT
< Content-Type: text/html; charset=utf-8
Content-Type: text/html; charset=utf-8

<
* Connection #0 to host 000000000000_123456.aws.qidi3dprinter.com left intact
```

Replicating the HTTP call of the QIDI Link mobile application



HTML page returned by AWS endpoint with invalid printer subdomain

Impact

All commands and responses to and from the printer traversing the internet may be sniffed and manipulated.

This could result in:
- Modifying printer.cfg and achieving arbitrary code execution (see Finding 1.)
- Alternative commands being executed
- Sensitive data such as webcam streams or model STLs being intercepted

Recommendation

- Immediate suspension of all QIDI Link functionality
- The total redesign of the remote connectivity function should consider:
  - ◇ Use modern, well-tested authentication mechanisms
  - ◇ Use high entropy secrets
  - ◇ Store and distribute said secrets through secure channels
  - ◇ Prefer anonymous identifiers such as UUIDv4 over predictable and trackable information such as MAC / IP address

Update 2024-10-01

UNMODIFIED

Client to frp server connection remains unencrypted (HTTP) as confirmed by the new frpc config generation logic and real world testing of the endpoint.

3. OTA Upgrade uses Insecure Protocol

CVSS 4.0

- **Severity**: High
- **Score**: 7.7
- **Vector**: CVSS:4.0/AV:N/AC:L/AT:P/PR:N/UI:P/VC:H/VI:H/VA:H/SC:N/SI:N/SA:N

Finding

The OTA firmware upgrade function communicates with the cloud endpoint over plaintext HTTP.

An attacker appropriately positioned would be able to falsify available updates and serve malicious firmware. The user would be prompted with a fake new version update, which, upon installation would permit arbitrary code execution as root on the target printer.

The OTA upgrade process works as follows (simplified):
1. Printer periodically issues HTTP GET to `http://www.aws.qidi3dprinter.com:5050/downloads/QD_Q1/version`
2. Remote version is compared to local version to establish if a newer version is available. If this is the case, the user is prompted to perform an upgrade.
3. Printer request firmware by issuing a HTTP GET to `http://www.aws.qidi3dprinter.com:5050/downloads/QD_Q1/V4.4.19/QD_Q1_SOC` (version = `Vx.y.z`, filename = (`QD_Q1_SOC` | `QD_Q1_UI`))
4. Received firmware files are just `.deb` packages are then installed unconditionally by `dpkg`

```
[...]
        base_download_url = "http://www.aws.qidi3dprinter.com:5050/downloads/QD_Q1"
[...]
    update_info_url = f"{base_download_url}/{version}/update_info.ini"
    destination_path = os.path.join(destination_dir, "update_info.ini")
    try:
        response = requests.get(update_info_url)
[...]
```

Excerpt from `/root/auto_update/version_check_Q1.py`

```
[...]
        base_download_url = "http://www.aws.qidi3dprinter.com:5050/downloads/QD_Q1"
[...]
url = f"{base_download_url}/{version}/{filename}"
    local_path = os.path.join(tmp_download_dir, filename)
    try:
        response = requests.get(url, stream=True)  # Ensure streaming download
[...]
```

Excerpt from `/root/auto_update/download_update_Q1.py`

Impact

An attacker could install an arbitrary package on the vulnerable printer if the user uses the OTA upgrade function.

This could result in:
- Loss of life - overriding klipper safeguards and commanding heaters (including the 240V mains heater) to run until something shorts or ignites
- Espionage through the onboard camera

• Theft of wifi credentials
• Theft of user files
• Enrolment into a botnet
• Pivotting to more sensitive devices on the LAN
• Denial of Service

Recommendation

• Immediate suspension of the OTA upgrade function
• Cease to use homebrew upgrade mechanisms where secure, well-tested alternatives exit such as `apt`

Update 2024-10-01

ELIMINATED

Managing updates appears to now be conducted over HTTPS with the `https://api.qidi3dprinter.com/code/q1_pro_version_info` and similar endpoints. The TLS certificate at the current time is valid and signed by a trusted CA (DigiCert Inc). As such I would consider this an effective mitigation of the above finding.

4. OTA Upgrade Lacks Integrity Verification

CVSS 4.0
• **Severity**: High
• **Score**: 7.7
• **Vector**: CVSS:4.0/AV:N/AC:H/AT:P/PR:N/UI:P/VC:H/VI:H/VA:H/SC:N/SI:N/SA:N

Finding
The OTA firmware upgrade function does not perform any integrity verification of received firmware before installing it.

An attacker able to serve malicious firmware to the printer could achieve arbitrary code execution as root as the printer does not perform any form of integrity checking. This means that unintentionally corrupted (in transit) or maliciously doctored firmware would be installed unconditionally.

```
[...]
        local_path = os.path.join(tmp_download_dir, component_names[component])
        install_cmd = f"dpkg -i --force-overwrite {local_path}"
        result = subprocess.run(install_cmd.split(), stdout=subprocess.PIPE,
stderr=subprocess.PIPE)
[...]
```

Excerpt from `/root/auto_update/download_update_Q1.py`

Impact
An attacker could install an arbitrary package on the vulnerable printer if the user uses the OTA upgrade function.

This could result in:
• Loss of life - overriding klipper safeguards and commanding heaters (including the 240V mains heater) to run until something shorts or ignites
• Espionage through the onboard camera
• Theft of wifi credentials
• Theft of user files
• Enrolment into a botnet
• Pivotting to more sensitive devices on the LAN
• Denial of Service

Recommendation
• Immediate suspension of the OTA upgrade function
• Cease to use homebrew upgrade mechanisms where secure, well-tested alternatives exist such as `apt`
• Sign all packages with gpg keys as per standard `apt` package lifecycle

Update 2024-10-01
REDUCED - Medium

Whilst the update debian package continues to be unconditionally applied with no integrity checking, the update is retrieved over TLS which does afford protection in transit. Nonetheless, the threat of untrusted updates being hosted on the server persists.

5. EoL Software Identified

CVSS 3.1
• **Severity**: High
• **Score**: 7.7
• **Vector**: CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:L

Finding
It was observed that an EoL version of nginx was running at `www.aliyun.qidi3dprinter.com`

The server header indicates version 1.20.1 is installed which has been End of Life for 2 years and is vulnerable to multiple CVEs e.g. https://nvd.nist.gov/vuln/detail/CVE-2021-23017

```
snow@kali ~/qidi $ curl http://www.aliyun.qidi3dprinter.com -I
HTTP/1.1 200 OK
Server: nginx/1.20.1
Date: Wed, 29 May 2024 23:36:25 GMT
Content-Type: text/html
Content-Length: 452
Last-Modified: Thu, 01 Jul 2021 09:21:13 GMT
Connection: keep-alive
ETag: "60dd8909-1c4"
Accept-Ranges: bytes
```
HTTP response of vulnerable server disclosing nginx version

Other web servers are in use across domains and reverse proxies but most have server header versions disabled (good practise) so it is difficult to determine if these are also EoL

I strongly suspect this is symptomatic of an endemic issue with poor patch management processes and this is merely the tip of the iceberg.

Impact
The vulnerable web server identified could be exploited using publicly available exploits.

The effects of this are widely dependent on the nature of the exploit but may include:
• Denial of Service
• Disclosure of customer information (accounts / email addresses / passwords / printer subdomains)
• Poisoning official firmware with malicious code
• Compromise of additional assets within the cloud VPC

Recommendation
• Update all software to the latest versions provided by the vendor
• Create / fix internal patch management processes to ensure a regular cadence of patching
• Proactively monitor external services for software approaching and breaching EoL

Update 2024-10-01
ELIMINATED

The server header now returns the unversioned response `nginx`. It is unknown if the backend servers were truly patched or simply obfuscated, but in absence of evidence to suggest otherwise, I will assume they were correctly updated.

6. SSH Permits Password Authentication

CVSS 4.0
• **Severity**: Info
• **Score**: 0
• **Vector**: N/A

Finding
It was identified that the cloud endpoints expose SSH publicly with the password authentication method permitted. This is default but not recommended.

Allowing username + password combinations for authentication is potentially dangerous if a weak password is chosen. An attacker could perform a bruteforce or dictionary attack to gain access to a shell on the remote endpoint.

```
Starting Nmap 7.95 ( https://nmap.org ) at 2024-05-30 01:38 BST
Nmap scan report for api.qidi3dprinter.com (54.68.120.215)
Host is up (0.18s latency).
rDNS record for 54.68.120.215: ec2-54-68-120-215.us-
west-2.compute.amazonaws.com

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.7 (protocol 2.0)
| ssh-auth-methods:
|   Supported authentication methods:
|     publickey
|     gssapi-keyex
|     gssapi-with-mic
|_    password
```
Supported authentication methods as reported by the endpoint

Impact
A weak password could result in an attacker guessing correctly and gaining managerial access to the cloud infrastructure.

Depending on configuration and hardening of assets, this may escalate into data loss and total control of the cloud infrastructure.

Recommendation
• Disable password based authentication and prefer publickey
• Use hardware tokens e.g. yubikeys
• Limit SSH access to trusted subnets such as a VPC
• Install and configure bruteforce countermeasures such as fail2ban

Update 2024-10-01
ELIMINATED

Password authentication has been disabled in preference of publickey and gssapi.

7. Test Pages Publicly Accessible

CVSS 4.0
• **Severity**: Info
• **Score**: 0
• **Vector**: N/A

Finding
Multiple test endpoints are publicly accessible which demonstrates poor environment segregation.

By contaminating production and development environments, there is a significantly elevated risk of vulnerabilities being introduced into production during development.

```
snow@kali ~/qidi $ curl https://api.qidi3dprinter.com/ -s
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Communication Test</title>
    <script type="text/javascript" src="./include/model.js"></script>
    <!-- <script>
        function sendMessageToCpp() {
            var message = {
                action: 'message',
                content: 'Hello from HTML!'
            };
            window.external.sendMessage(JSON.stringify(message));
        }

        function receiveMessageFromCpp(message) {
            var output = document.getElementById('output');
            var parsedMessage = JSON.parse(message);
            output.innerHTML = 'Received message from C++: ' +
parsedMessage.content;
        }
    </script> -->
</head>
<body>
    <h2>Communication Test</h2>
    <button onClick="OpenWikiUrl('https://wiki.bambulab.com/en/software/bambu-
studio/studio-quick-start')">Send Message to C++</button>
    <div id="output"></div>
</body>
</html>
```

Excerpt from a test API endpoint appearing to plagiarise functionality from Bambulab printers

Impact
Potential unnecessary introduction of additional vulnerabilities due to test interfaces.

Recommendation
• Strictly segregate environments between development and production
• Create processes for promoting material from dev to prod
• Use version control internally to permit logging and rollbacks

Update 2024-10-01
ELIMINATED

8. Security Critical Code Obfuscated

CVSS 4.0
• **Severity**: Info
• **Score**: 0
• **Vector**: N/A

Finding
It was noted that the python file `/home/mks/qrcode/qrcode_QD.py` has been heavily obfuscated with pyarmor V8.

Aside from violating the license agreement of the shareware package pyarmor, it is concerning that code which is responsible for securely transmitting sensitive data (the subdomain string) has been intentionally obfuscated.

Whilst the deterrent effect to plagiarism is evident, obfuscating code does not provide any security. On the contrary, it makes it significantly harder to audit whilst providing little resistance to a dedicated actor.

Preliminary code profiling doesn't indicate anything malicious, however it was not possible to entirely rule it out due to time constraints.

```
# Pyarmor 8.4.7 (trial), 000000, non-profits, 2024-04-09T10:56:18.193539
from pyarmor_runtime_000000 import __pyarmor__
__pyarmor__(__name__, __file__,
b'PY000000\x00\x03\x07\x00B\r\r\n\x80\x00\x01\x00\x08\x00\x00\x00\x04\x00\x00\
x00@\x00\x00\x00\xee\x19\x00\x00 [...]
```

Impact
None

Recommendation
• Stop obfuscating security critical code
• Respect license agreements

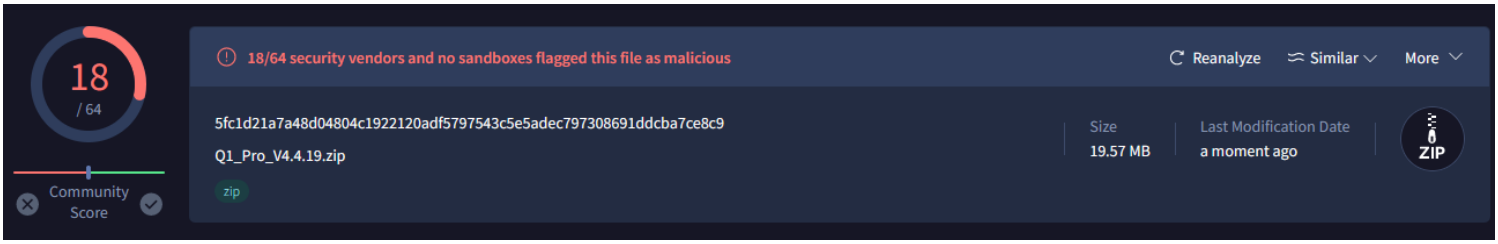
Update 2024-10-01
UNCHANGED

With the move from python scripts to the compiled xindi binary, security critical code has become even harder to review. Thankfully, pyarmor is no longer being used against its license agreement.

9. Firmware Releases Flagged as Malware

CVSS 4.0
- **Severity**: Info
- **Score**: 0
- **Vector**: N/A

Finding
Firmware releases of the Q1 Pro firmware ≥ 4.4.16 are flagged by multiple antivirus suites as containing a trojan or reverse proxy. Further investigation determined this to be a false positive.



The culprit file is `root/frp/frpc` with sha-256 sum `6ce02ed8d82ce546cef85fa57cf66df1eea732da2ffd3a4528f46c0304eca2e7` a.k.a. Fast Reverse Proxy Client. This is same binary as in the official project release `v0.53.20` (link) (arch = `arm64`) implying Qidi have included an officially compiled version and ruling out the possibility of code alterations.

The reason this is flagged by antivirus software is because this binary has been observed to be used by real threat actors to establish a reverse tunnel during attacks. The binary itself is benign.

Impact
None

Recommendation
Despite this being a false positive, it is imperative to reinforce to not disregard antivirus alerts and only permit exceptions if there is not doubt regarding the nature of the file. It would be better for Qidi to clearly communicate exactly why this is a false positive. Claiming it is needed by QIDI Link, whilst true, does not suffice.

Update 2024-10-01
UNCHANGED

The same binary is bundled with the update rendering the above finding unchanged. To my knowledge, no clarifying statements have been released to explain the reason for the false positive beyond "it's needed by QIDI Link".