Investigating the Components of Zero-Trust Networking at Home

Nicholas Anthony
CS 434: Systems and Network Security II
Professor Dan Carrere
June 11, 2024

Table of Contents

Abstract

Over the last decade, the security perimeter of enterprise networks has expanded rapidly. As working from home and remote VPN connectivity have become common in the workplace, current network security has struggled to cope with the renewed threat of insider attacks and phishing attempts. As a stopgap solution to prevent more attacks of this nature from taking place, zero-trust networking (ZTN) concept integration is now one of the most popular and talked-about enterprise security solutions. However, this expanded security perimeter is not confined to enterprise. At-home users, who have increasing numbers of IoT devices with weak security, are also vulnerable. As this paper will show, zero-trust networking concepts can also be beneficial to at-home users who wish to keep their homes safe from lateral and vertical privilege escalation stemming from compromised IoT devices with weak security, without needing to spend an excess of funds on consumer products (or otherwise) beyond the scope of an average at-home networking setup.

Introduction

Zero-trust networking (ZTN) is a concept that many in the security space lament is open-ended and often ambiguous, so for the purposes of this project I will attempt to clearly define what is meant by zero-trust. Many use the phrase “zero-trust network access” to define the category of technologies that provide secure remote access to resources based on access control

policies. In this project, I will generalize to just “zero-trust” as a holistic approach that incorporates and heightens several key security concepts, as described by Cloudflare: continuous monitoring and validation, least privilege, device access control, microsegmentation, lateral movement prevention, and multi-factor authentication [1]. Essentially, zero-trust is exactly that: viewing trust as something to be provisioned according to use-case, instead of something given by default as with many networks.

In an enterprise environment, implementing ZTN access extends existing secure access control measures, permitting relatively worry-free connection from anywhere. It’s relatively scalable, meaning that restrictive access to non-company managed devices (e.g an external firm’s laptops) can be provided quickly. The reason why ZTNA has spread like wildfire so quickly is in the effect of remote access on enterprise security perimeters, along with insider threats. Consistent and constant monitoring and verification can help identify any insiders who seek to steal or damage company property, or the lateral/vertical movements of compromised internal machines (perhaps through a phishing-type attack). There are many instances of attacks that could have been better mitigated or prevented through effective implementation of ZTN concepts. One such example is the 2014 Sony Pictures hack, in which the entire Sony network was infected (and subsequently destroyed) through just a single compromised machine [2]. Microsegmentation, proper monitoring, and lateral movement prevention are all ZTN concepts that could have helped.

However, one may wonder what the benefits are of a zero-trust model for home networks. Many assume that home networks are safe as soon as a router is plugged in and a password is set – and they’re not totally wrong – but a recent technology threatens to eliminate such base security: IoT. The world observed in 2016 the carnage wrought by the Mirai botnet [3],

which comprised hundreds of thousands of compromised IoT devices, from cameras to fridges to garage door openers. IoT security is a relatively new and evolving field, but most devices being shipped out to consumers today still suffer from inconsistent security standards, lack of encryption, spotty firmware updates, limited device management, and physical vulnerabilities [3]. IoT devices can be the perfect foothold for potential attackers looking to escalate privilege on home networks, who thus far have mostly focused on leveraging the IoT devices network capabilities for DDoS attacks [3]. Therefore, it would be fair to say that many home users with IoT devices on their networks can assume that there is a chance they may already be compromised. In this case, where attacks from the inside of the network are possible, implementing zero-trust security concepts is an effective strategy to counter the threat from IoT vulnerabilities. I chose this project precisely because there is a lack of interest and information for consumers in this area.

Project Concept and Methodology

In this project, I will look at zero-trust networking concepts from the point of view of a normal home user, who does not wish to purchase any additional networking devices for their home. Generally, most home networks consist of a modem, router, and connected devices such as laptops, tower computers, gaming consoles, televisions, phones, and other various IoT technologies. Many consumer routers do not have the capability out of the box to effectively implement zero-trust networking concepts. Thankfully, many routers are able to host OpenWrt, a Linux-based firmware that provides more comprehensive customization and additional network packages. For this project, I demonstrate and construct a network that is a slight augmentation

from a very common home network, to show that home users can make their networks safe from insider threats with relatively little (guided) effort.

Home users, if interested in bolstering their own security, still may be unfriendly towards concepts like MFA and continuous verification, as they are invasive and persistent. Thus, zero-trust concepts that will be implemented in this project are as follows: microsegmentation, least privilege, continuous monitoring, and lateral movement prevention.

Microsegmentation

There are a couple of ways that users can segment their networks, the most common being separate SSIDs or VLANs. In this project, I am choosing to implement separate SSIDs, because they are the easiest for a home user to understand: separate networks from the same router. I am segmenting my network into three categories: IoT network, guest network, and authorized user network. Each of these networks will not be able to see the other and the ESSIDs will be hidden to reduce the efficacy of attacker network scans.

Least Privilege & Lateral Movement Prevention

The principle of least privilege states that users/devices should be allotted only as much privilege as they need to perform their necessary tasks - no more, no less. In this way, traffic from the guest network is restricted to ports 80 and 443, typical web application (HTTP/HTTP/s) protocols. It is a little harder to restrict IoT ports, because each IoT developer may choose different ports for communications, and some devices use different protocols like Bluetooth or Zigbee. Additionally, clients on the IoT and guest networks are isolated to prevent any traffic sniffing, MitM attacks, or lateral movement.

Continuous Monitoring

Logs on the router are only viewable traditionally through the router dashboard, and are relatively non verbose compared to the viewable information that a router can handle. To remedy this, a syslog forwarder will forward all logs to a syslog server to the network operators' machine. On top of that, fail2ban and Crowdsec will contribute towards perimeter monitoring.

Implementation

For a demonstration and example setup of the network architecture, I used a NetGear (R6120) AC1200 Dual Band Router. Following OpenWrt's wiki, NetGear has a special protocol called NMRP that allows for TFTP communication (presumably to fetch new firmware images). Using this protocol, home users can flash any firmware they desire onto their NetGear routers using a tool called nmrpflash. OpenWrt images exist for hundreds of different routers and have similar setups to this NetGear router.

Flashing the Firmware

1. First, I connected the router and my PC through a wired Ethernet connection.
2. Then, I set a static IP on my machine, so that a stable connection is maintained throughout the file transfer.

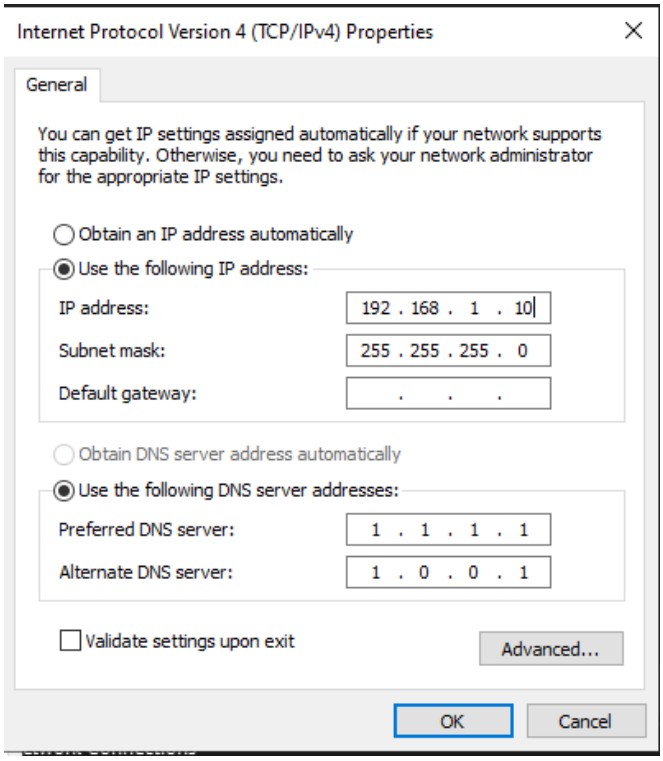

Fig.1 - Setting the static IP of my machine

3. After that, I hosted a TFTP server on my machine and had it point to the directory containing nmrpflash and the OpenWrt firmware image. Then, I turned off the router, started nmrpflash, and turned the router back on.

```
Microsoft Windows [Version 10.0.19045.2546]
(c) Microsoft Corporation. All rights reserved.

C:\Users\nick>cd Desktop

C:\Users\nick\Desktop>nmrpflash -L
net9   192.168.1.3      74:56:3c:4a:e2:bc  (Ethernet)

C:\Users\nick\Desktop>
```

Fig.2 - nmrpflash listing the network interfaces

```
C:\Users\nick\Desktop>nmrpflash -i net9 -f openwrt-ramips-mt76x8-netgear_r6120-squashfs-factory.img
Advertising NMRP server on net9 ... \
Received configuration request from 80:cc:9c:bc:0d:0a.
Sending configuration: 10.164.183.253/24.
Received upload request without filename.
Uploading openwrt-ramips-mt76x8-netgear_r6120-squashfs-factory.img ... OK (5577299 b)
Waiting for remote to respond.
Received keep-alive request (3).
Remote finished. Closing connection.
Reboot your device now.
```

Fig. 3 - flashing the firmware to the router

4. Internet connectivity is not established yet, so I needed to ssh into the router in order to reset all settings, update the package manager and install the web interface.

```
C:\Users\nick>ssh root@192.168.1.1
The authenticity of host '192.168.1.1 (192.168.1.1)' can't be established.
RSA key fingerprint is SHA256:LqlrE3rVlmeIDisdDZrxlYVCfhlROlKmUdpgzJYWb2k.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.1.1' (RSA) to the list of known hosts.


BusyBox v1.36.1 (2024-06-05 16:44:11 UTC) built-in shell (ash)

  _______                     ________        __
 |       |.-----.-----.-----.|  |  |  |.----.|  |_
 |   -   ||  _  |  -__|     ||  |  |  ||   _||   _|
 |_______||   __|_____|__|__||________||__|  |____|
          |__| W I R E L E S S   F R E E D O M
 -----------------------------------------------------
 OpenWrt SNAPSHOT, r26524-3de3c2bdfa
 -----------------------------------------------------
=== WARNING! =====================================
There is no root password defined on this device!
Use the "passwd" command to set up a new password
in order to prevent unauthorized SSH logins.
--------------------------------------------------
root@OpenWrt:~# jffs2reset
This will erase all settings and remove any installed packages. Are you sure? [N/y]
y
/dev/mtdblock6 is mounted as /overlay, only erasing files
root@OpenWrt:~#
```

Fig. 4 - ssh into the router

```
root@OpenWrt:~# opkg update
Downloading https://downloads.openwrt.org/snapshots/targets/ramips/mt76x8/packages/Packages.gz
Updated list of available packages in /var/opkg-lists/openwrt_core
Downloading https://downloads.openwrt.org/snapshots/targets/ramips/mt76x8/packages/Packages.sig
Signature check passed.
Downloading https://downloads.openwrt.org/snapshots/packages/mipsel_24kc/base/Packages.gz
Updated list of available packages in /var/opkg-lists/openwrt_base
Downloading https://downloads.openwrt.org/snapshots/packages/mipsel_24kc/base/Packages.sig
Signature check passed.
Downloading https://downloads.openwrt.org/snapshots/targets/ramips/mt76x8/kmods/6.6.32-1-2848dc58187556a548aac07e4109fdf
d/Packages.gz
Updated list of available packages in /var/opkg-lists/openwrt_kmods
Downloading https://downloads.openwrt.org/snapshots/targets/ramips/mt76x8/kmods/6.6.32-1-2848dc58187556a548aac07e4109fdf
d/Packages.sig
Signature check passed.
Downloading https://downloads.openwrt.org/snapshots/packages/mipsel_24kc/luci/Packages.gz
Updated list of available packages in /var/opkg-lists/openwrt_luci
Downloading https://downloads.openwrt.org/snapshots/packages/mipsel_24kc/luci/Packages.sig
Signature check passed.
Downloading https://downloads.openwrt.org/snapshots/packages/mipsel_24kc/packages/Packages.gz
Updated list of available packages in /var/opkg-lists/openwrt_packages
Downloading https://downloads.openwrt.org/snapshots/packages/mipsel_24kc/packages/Packages.sig
Signature check passed.
Downloading https://downloads.openwrt.org/snapshots/packages/mipsel_24kc/routing/Packages.gz
```

Fig. 5 - Updating the package manager

5. After rebooting once more, the web interface is now viewable.

This was the specific setup for my router, but all subsequent steps I generalized for home users looking to implement my same setup.

Setting up the SSIDs

1. Navigate to the drop down menu and select Network - > Wireless.
2. Guest network/IoT network (repeatable for each)
    a. Select the “Add” button under the 2.5 GHz radio (or lower band)
    b. Enable Wireless, and under Interface Configurations add ESSID, and select “Hide ESSID”, also select “WMM mode.”
    c. Under “Wireless Security” enable WPA2-PSK and choose a password, then enable KRACK countermeasures.
    d. Keep the MAC address filter disabled.
        i. For the IoT network, enable the MAC address filter and add the MAC addresses of your IoT devices.
    e. Under “Advanced Settings” select “Isolate Clients”
    f. In Configuration Advanced Settings, select the country code.
3. Authorized users network
4. Select the “Add” button under the 5 GHz radio
5. Enable Wireless, and under Interface Configurations add ESSID, and select “Hide ESSID”, also select “WMM mode.”

6. Under “Wireless Security” enable WPA3-SAE and choose a password, then enable KRACK countermeasures.
7. Add MAC addresses of authorized devices.
8. In Configuration Advanced Settings, select the country code.

Firewall Rules

First, we need to create a new firewall zone for the guest network.

1. Navigate to Network -> Firewall -> General Settings
2. Click “Add” and then add the guest network AP under “covered devices”
3. Set the Input, Output, and Forward options to “Accept”
4. Name it something memorable, ex: “Guest”

Now that the zone is created for the guest network, we can apply firewall rules to it.

1. Navigate to Traffic Rules and select “Add” -> Allow-HTTPS and apply the following settings:
    a. Protocol -> TCP
    b. Source zone -> “Guest”
    c. Destination zone -> wan
    d. Destination port -> 443
2. Repeat with Allow-HTTP (just replace the port with 80)
3. Then we will disallow all other traffic. Add another traffic rule called “Block-others”
    a. Protocol -> Any
    b. Source zone -> “Guest”

c. Destination zone -> wan

d. Action -> “drop”

4. It is crucial that the “allow” rules are listed before the drop rule.

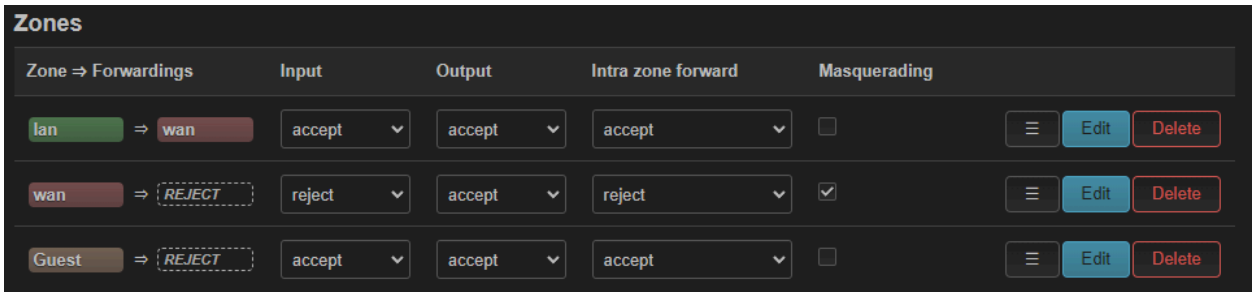

Fig. 6 - Firewall zones (note “Guest”)

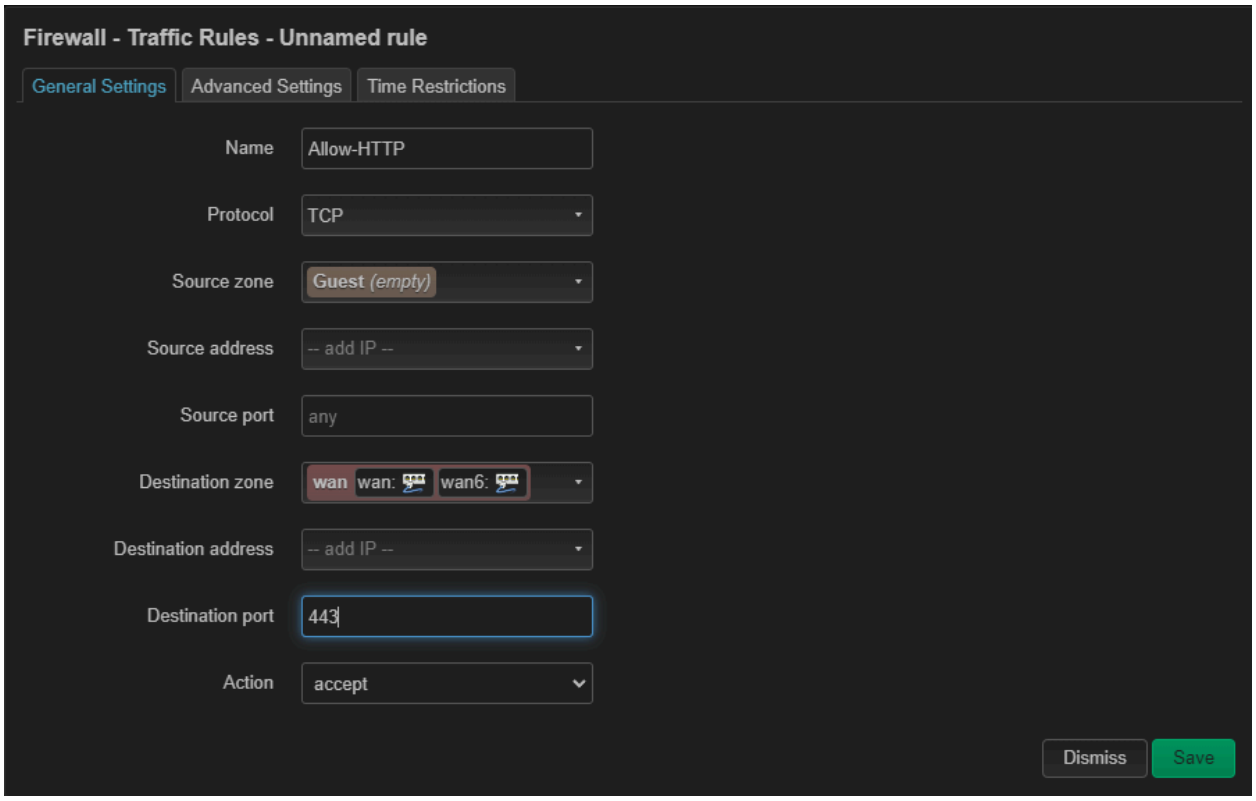

Fig. 7 - Example firewall rule allowing traffic on 443 on the Guest network

System log forwarding

Myriad issues occurred when I tried to get more comprehensive logging software to work (such as Splunk), but there is a lightweight system logging software available for OpenWrt: syslog-ng. Installing is simple: one must only run *opkg install syslog-ng* on the OpenWrt system (after SSHing in of course) and then in /etc/syslog-ng.conf setting the port and IP of the machine running the syslog server.

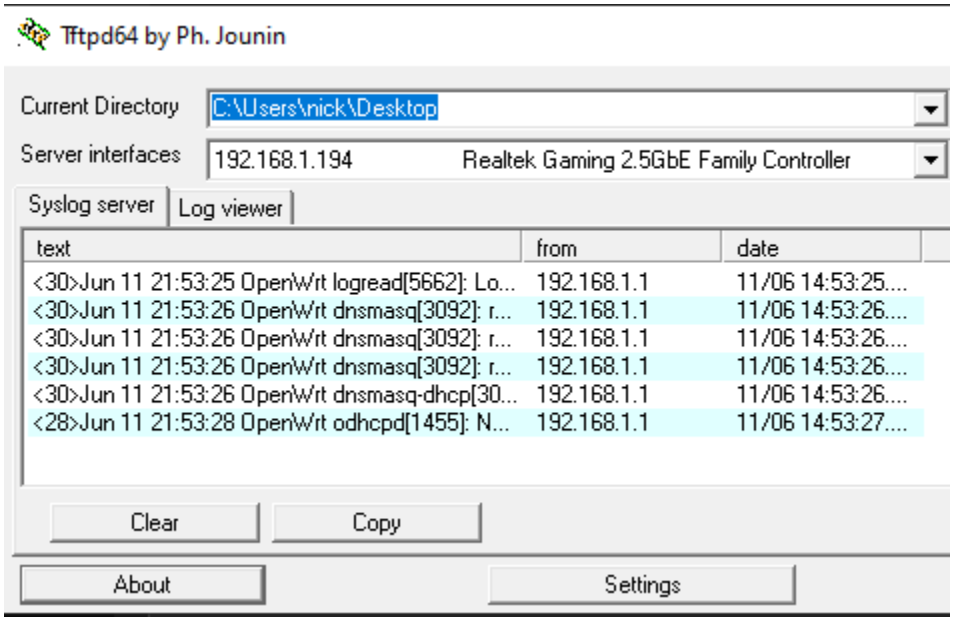

Fig. 8 - Syslog input from the OpenWrt router on my local PC

Installing packages

Installing packages was simple, simply run *opkg install fail2ban* and *opkg install crowdsec*. If there is not enough storage on the router, simply plug in and mount a USB to the router (most have at least one USB 2.0 or higher slot). I configured fail2ban to enable an SSH jail and incremental bans for repeat offenders, but network operators can deal with specific configuration at their discretion.

Results and Future Considerations

The resulting network topography can be viewed in Fig 9. As a result, this network is much more secure from insider (IoT) threats and correctly utilizes ZTN concepts.

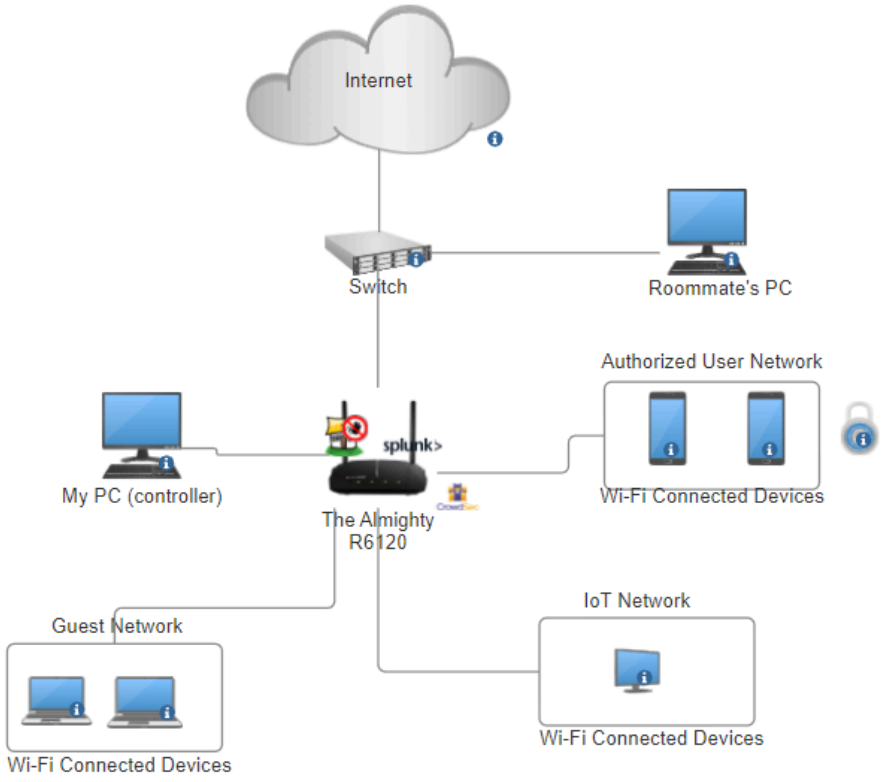

Fig. 9 - Resulting network topography

For future considerations, one very big idea I have is a script that automatically detects router models and flashes OpenWrt automatically, even running an Ansible script to download LuCI and configure all of the networks. This way, home users would only need to follow 2 or 3 steps to achieve a much more secure and segmented network. Additionally, better tools could be implemented into the flow (like an application to consolidate all the logs from the various packages on the router).

References

[1] “What Is Zero Trust Security? Principles of the Zero Trust Model.” *Crowdstrike.Com*, 22 Jan. 2024, www.crowdstrike.com/cybersecurity-101/zero-trust-security/

[2] *The Hack on Sony Group Pictures Entertainment*, www.secureops.com/wp-content/uploads/2021/06/Sony-Breach-Analysis-v4.pdf. Accessed 9 June 2024.

[3] Henke, Christian. “IOT Security: Risks, Examples, and Solutions: IOT Glossary.” *IoT Security: Risks, Examples, and Solutions | IoT Glossary*, EMnify GmbH, 5 Mar. 2024, www.emnify.com/iot-glossary/iot-security.

[4] Chubb, Peter. *NMRP --- Unbrick Protocol for NetGear Routers*, web.archive.org/web/20240315142516/www.chubb.wattle.id.au/PeterChubb/nmrp.html. Accessed 11 June 2024.