Upon receiving the firmware files (wr-841n.bin, dcs-8000lh.bin, bare-metal-demo.elf) for this project, the first course of action that was taken was to perform reconnaissance on them, which consisted of searching the internet for product information, such as who developed the product, what the product is, when it was released, and how the device can be obtained or purchased. After gathering this initial information, we conducted further research on the known vulnerabilities to determine what type of exploitable information could be obtained from them.

The first file we performed recon on was the wr-841n.bin, which is for the TP-Link TL-WR841N router. We discovered that this device can no longer be purchased new through Amazon, but it can be purchased used. In addition, we were also able to gather the types of Wifi modes it supports, the security protocols it uses, hardware specifications, User Manual (PDF), Installation Manual (PDF), and other basic release information details. Following our initial product information gathering, we performed a Google search for "TP-Link TL-WR841N vulnerabilities." This search yielded numerous websites detailing known weaknesses within the device's firmware. The most significant weakness is the dropbearpwd Improper Authentication Information Disclosure vulnerability, which allows attackers to expose sensitive information in the httpd service. This information provided a good roadmap as to what we could disclose with a successful attack. Upon further research, we were always able to determine that the "TP-Link" brand has had other devices with known vulnerabilities. In addition, at the beginning of 2025, a group of U.S. lawmakers/senators has requested that the "TP-Link" brand be banned in the United States due to the brand being closely tied to the Chinese Communist Party, and for suspicions of embedding surveillance capabilities into U.S. networks.
"https://industrialcyber.co/critical-infrastructure/us-lawmakers-push-to-ban-tp-link-over-national-security-risks-surveillance-concerns/#:~:text=In%20a%20bicameral%20letter%2C%20the,harmful%20capabilities%20into%20U.S.%20networks."

We also used noseyparker, which complements tools such as binwalk, file, grep, trufflehog, detect-secrets, etc. This software was able to detect hardcoded passwords, base64 credentials, default configuration passwords, and usernames.

Analysis for wr-841n.bin:
File ID: file returned data
Key findings:

- U-Boot string at 0xD120 U-Boot 1.1.3
- LZMA compressed kernel at 0x10200
- SquashFS root filesystem at 0x100000
- Header pattern: repeating 16-byte blocks at file start — likely firmware table/metadata
- Extraction status: binwalk -Me found components but warned sasquatch extractor missing; some symlinks rewritten for safety during extraction
- Immediate IOCs: MD5 97710bf8ae216d4665c1c89a43eca626; timestamps indicate build date 2022-08-16

- Risk/impact notes: contains full root filesystem — may include credentials, web UI, and device configs (sensitive if exposed)
- Tools used: file, xxd, strings, binwalk, binwalk -Me,
- Nosey Parker used and ran a report that discovered the squashfs-root file system. It found API keys for AWS and GitHub aligned with private keys and tokens. It also found the username of the admin and the password in base 64.

Figure 1.1

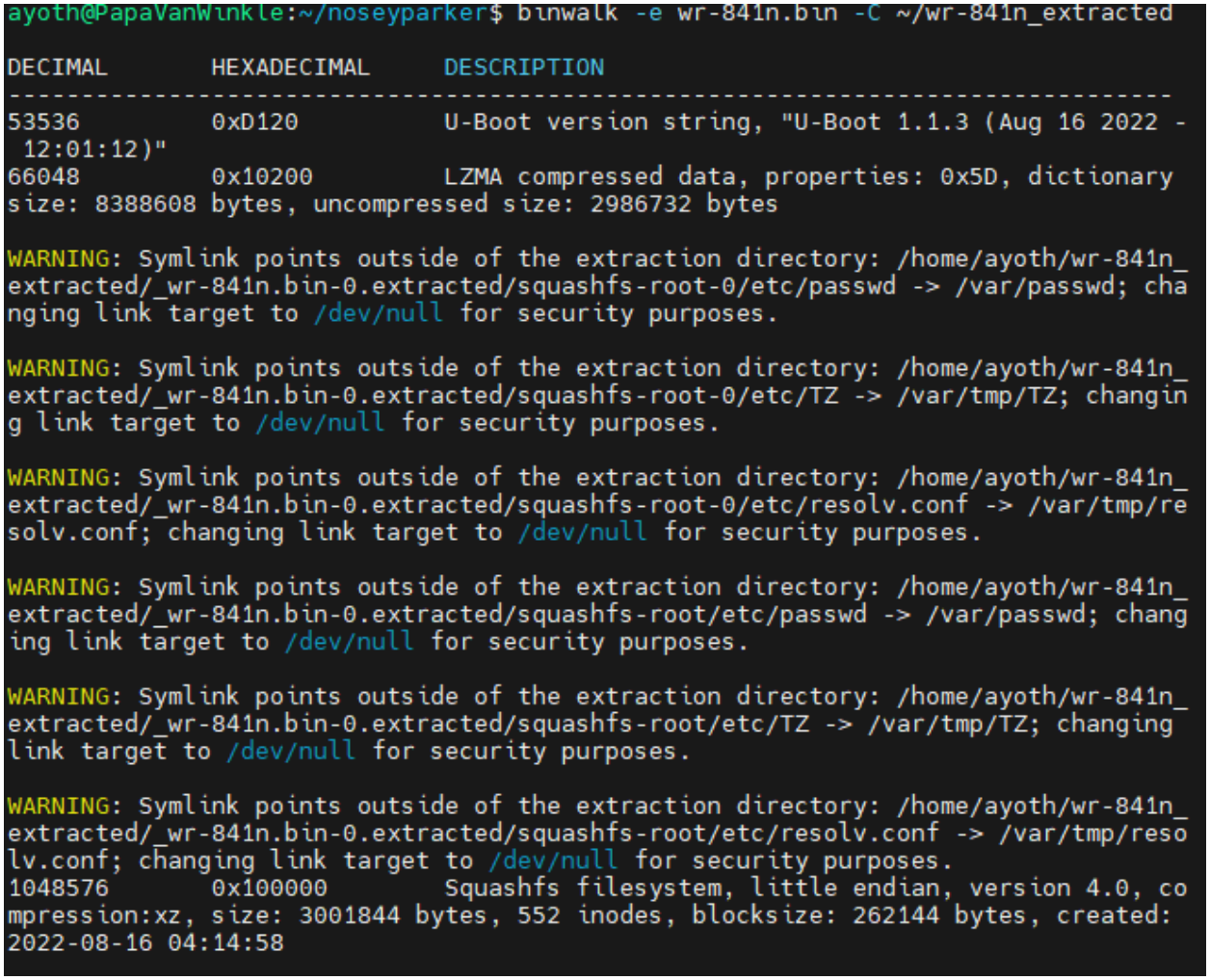


Figure 1.2

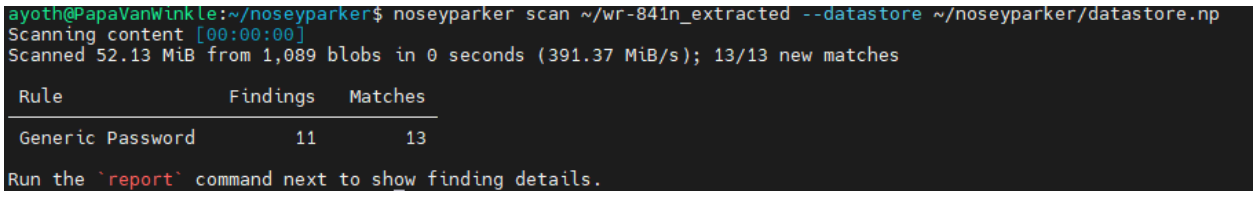


Figure 1.3

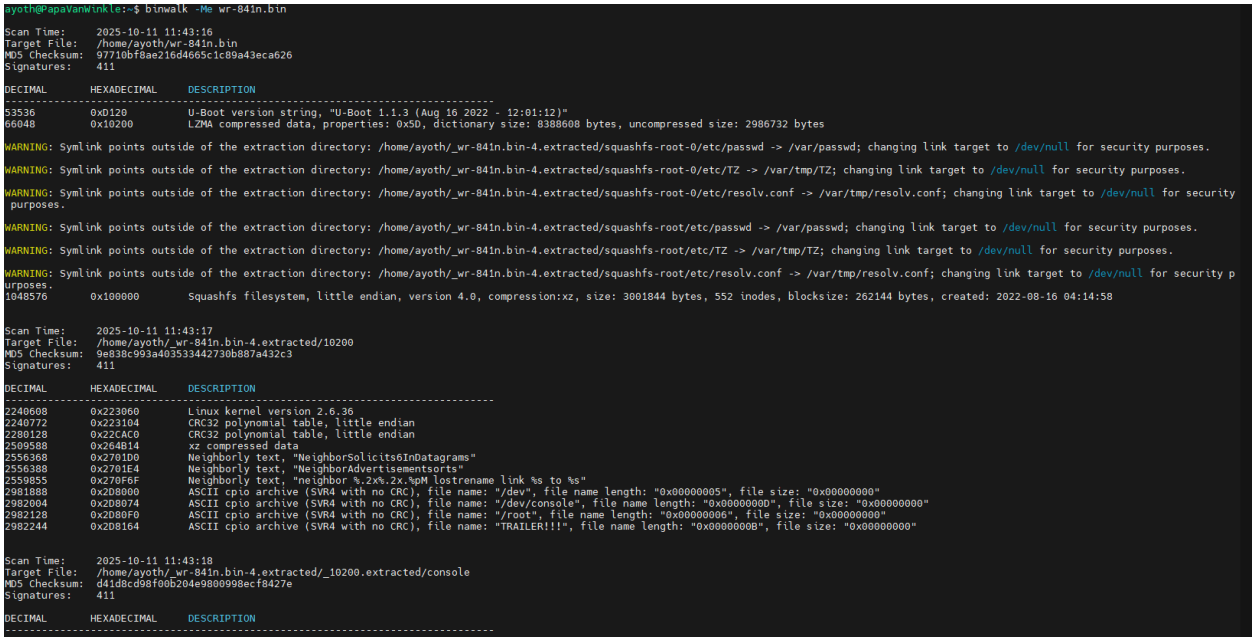


Figure 1.4

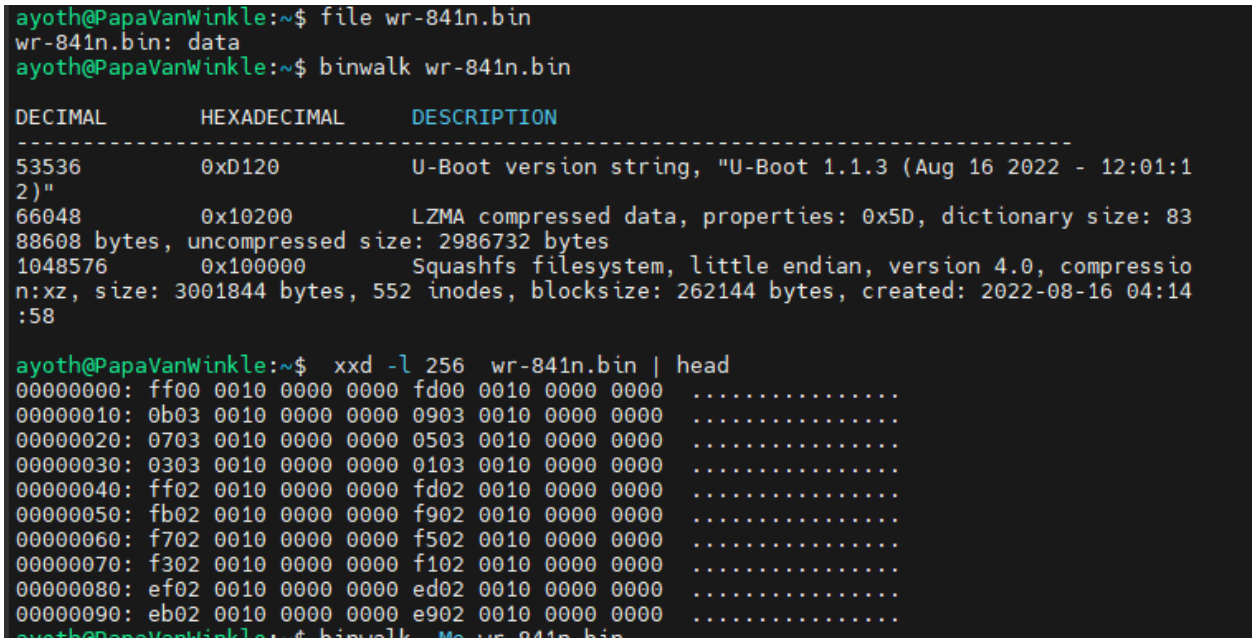


Figure 1.5

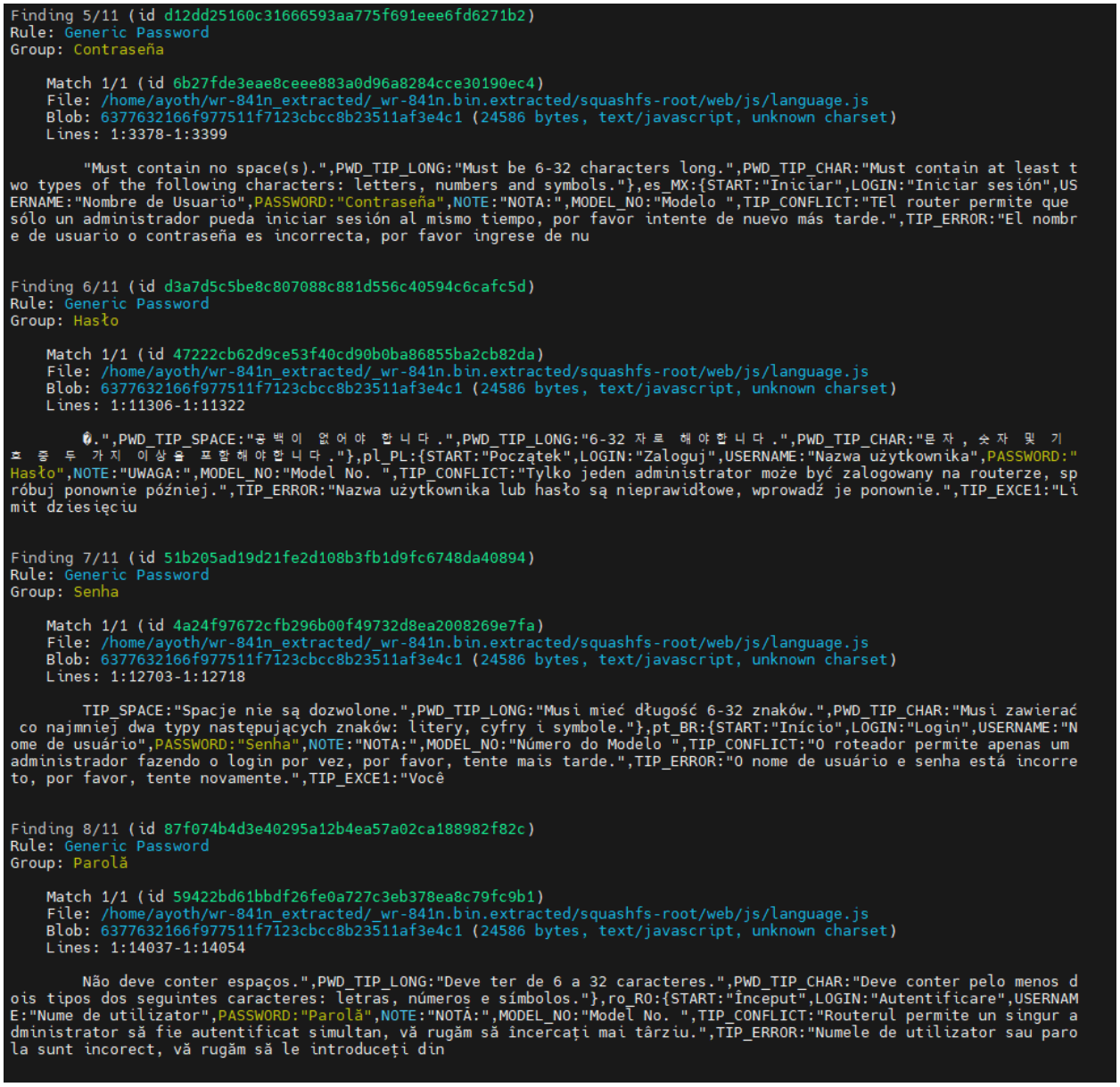
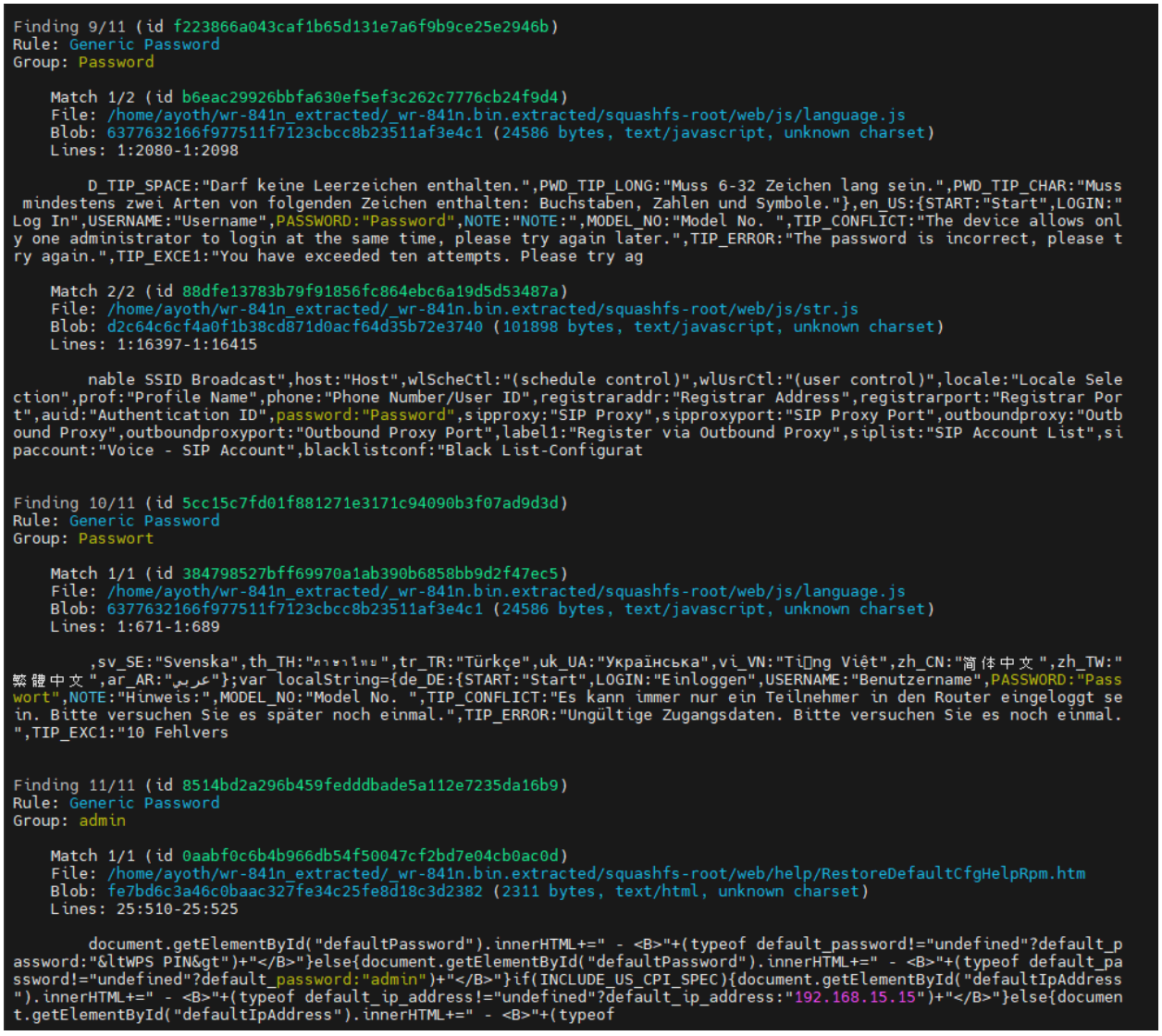

Finding 5/11 (id d12dd25160c31666593aa775f691eee6fd6271b2)
Rule: Generic Password
Group: Contraseña
Match 1/1 (id 6b27fde3eae8ceee883a0d96a8284cce30190ec4)
File: /home/ayoth/wr-841n_extracted/_wr-841n.bin.extracted/squashfs-root/web/js/language.js
Blob: 6377632166f977511f7123cbcc8b23511af3e4c1 (24586 bytes, text/javascript, unknown charset)
Lines: 1:3378-1:3399
"Must contain no space(s).",PWD_TIP_LONG:"Must be 6-32 characters long.",PWD_TIP_CHAR:"Must contain at least t
wo types of the following characters: letters, numbers and symbols."},es_MX:{START:"Iniciar",LOGIN:"Iniciar sesión",US
ERNAME:"Nombre de Usuario",PASSWORD:"Contraseña",NOTE:"NOTA:",MODEL_NO:"Modelo ",TIP_CONFLICT:"TEl router permite que
sólo un administrador pueda iniciar sesión al mismo tiempo, por favor intente de nuevo más tarde.",TIP_ERROR:"El nombr
e de usuario o contraseña es incorrecta, por favor ingrese de nu
Finding 6/11 (id d3a7d5c5be8c807088c881d556c40594c6cafc5d)
Rule: Generic Password
Group: Hasło
Match 1/1 (id 47222cb62d9ce53f40cd90b0ba86855ba2cb82da)
File: /home/ayoth/wr-841n_extracted/_wr-841n.bin.extracted/squashfs-root/web/js/language.js
Blob: 6377632166f977511f7123cbcc8b23511af3e4c1 (24586 bytes, text/javascript, unknown charset)
Lines: 1:11306-1:11322
.",PWD_TIP_SPACE:"공 백 이 없 어 야 합 니 다 .",PWD_TIP_LONG:"6-32 자 로 해 야 합 니 다 .",PWD_TIP_CHAR:"문 자 , 숫 자 및 기
호 중 두 가 지 이 상 을 포 함 해 야 합 니 다 ."},pl_PL:{START:"Początek",LOGIN:"Zaloguj",USERNAME:"Nazwa użytkownika",PASSWORD:"
Hasło",NOTE:"UWAGA:",MODEL_NO:"Model No. ",TIP_CONFLICT:"Tylko jeden administrator może być zalogowany na routerze, sp
róbuj ponownie później.",TIP_ERROR:"Nazwa użytkownika lub hasło są nieprawidłowe, wprowadź je ponownie.",TIP_EXCE1:"Li
mit dziesięciu
Finding 7/11 (id 51b205ad19d21fe2d108b3fb1d9fc6748da40894)
Rule: Generic Password
Group: Senha
Match 1/1 (id 4a24f97672cfb296b00f49732d8ea2008269e7fa)
File: /home/ayoth/wr-841n_extracted/_wr-841n.bin.extracted/squashfs-root/web/js/language.js
Blob: 6377632166f977511f7123cbcc8b23511af3e4c1 (24586 bytes, text/javascript, unknown charset)
Lines: 1:12703-1:12718
TIP_SPACE:"Spacje nie są dozwolone.",PWD_TIP_LONG:"Musi mieć długość 6-32 znaków.",PWD_TIP_CHAR:"Musi zawierać
co najmniej dwa typy następujących znaków: litery, cyfry i symbole."},pt_BR:{START:"Início",LOGIN:"Login",USERNAME:"N
ome de usuário",PASSWORD:"Senha",NOTE:"NOTA:",MODEL_NO:"Número do Modelo ",TIP_CONFLICT:"O roteador permite apenas um
administrador fazendo o login por vez, por favor, tente mais tarde.",TIP_ERROR:"O nome de usuário e senha está incorre
to, por favor, tente novamente.",TIP_EXCE1:"Você
Finding 8/11 (id 87f074b4d3e40295a12b4ea57a02ca188982f82c)
Rule: Generic Password
Group: Parolă
Match 1/1 (id 59422bd61bbdf26fe0a727c3eb378ea8c79fc9b1)
File: /home/ayoth/wr-841n_extracted/_wr-841n.bin.extracted/squashfs-root/web/js/language.js
Blob: 6377632166f977511f7123cbcc8b23511af3e4c1 (24586 bytes, text/javascript, unknown charset)
Lines: 1:14037-1:14054
Não deve conter espaços.",PWD_TIP_LONG:"Deve ter de 6 a 32 caracteres.",PWD_TIP_CHAR:"Deve conter pelo menos d
ois tipos dos seguintes caracteres: letras, números e símbolos."},ro_RO:{START:"Început",LOGIN:"Autentificare",USERNAM
E:"Nume de utilizator",PASSWORD:"Parolă",NOTE:"NOTĂ:",MODEL_NO:"Model No. ",TIP_CONFLICT:"Routerul permite un singur a
dministrator să fie autentificat simultan, vă rugăm să încercați mai târziu.",TIP_ERROR:"Numele de utilizator sau paro
la sunt incorect, vă rugăm să le introduceți din

Figure 1.6



The second file we performed recon on was the dcs-8000lh.bin, which is for the D-Link Mini HD WiFi Camera. In comparison to the TP-Link router, this device is still actively sold through Amazon.com and can be purchased for $18.95. Similarly, we were able to locate product information on this device, such as the flash memory type, power source, internet connectivity protocol, video capture format, and User Manual (PDF). Following our initial product information gathering, we performed a Google search for "D-Link dcs-8000lh Vulnerabilities." This search resulted in various vulnerabilities and also other known D-Link products that have the same security weakness as the 8000LH. The vulnerability these devices face consists of attackers being able to view the device's configuration file by running the following command: "<Camera-IP>/common/info.cgi" By doing so, attackers can get ahold of the following information from the device: model, brand, version, build, hw_version, nipca_version, device name, location, MAC address, IP address, gateway IP address, wireless status, input/output settings, speaker, and

sensor settings https://nvd.nist.gov/vuln/detail/cve-2018-18441#match-14738767. This information would allow for targeted attacks and comes with a low barrier of entry, given that all the attacker would need is the device's IP address. Once that has been obtained, the ability for the device to be immediately controlled by an attacker is significant. After running noseyparker on the DCS-8000lh.bin file, a 2048-bit RSA private key was found along with a https/tls certificate private key, unencrypted with no password protection. This exposes the camera to certificate authentication whomever and camera streams can be accessed.

Analysis for dcs-8000lh.bin

- A 2048-bit RSA private key was found along with a https/tls certificate private key, unencrypted with no password protection.

Figure 2.1

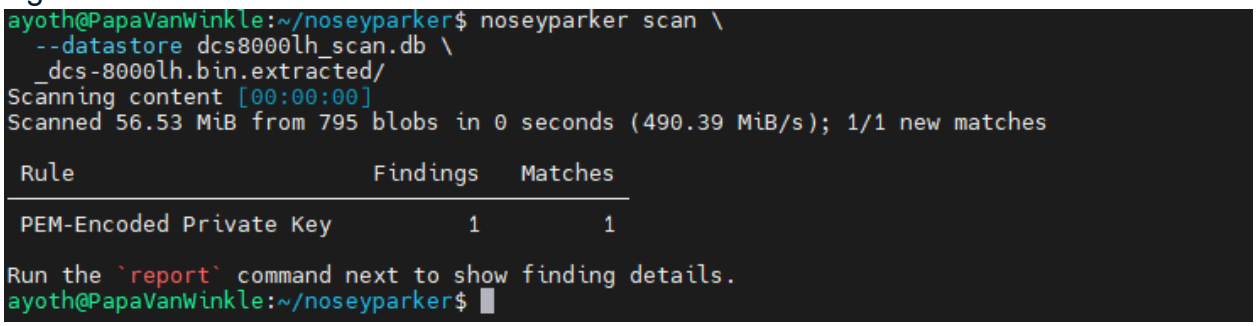


Figure 2.2

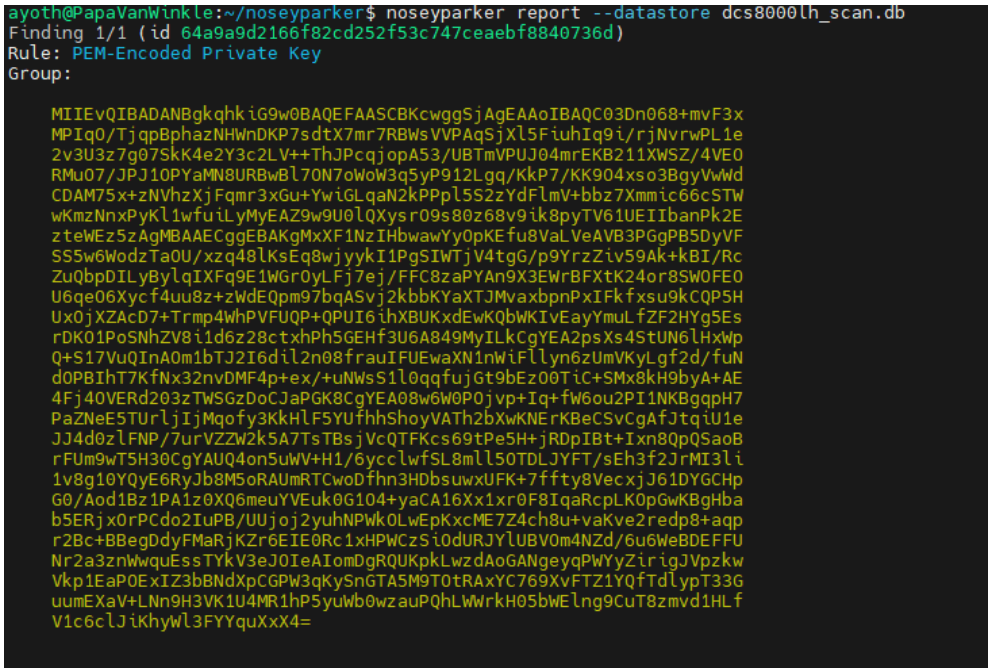
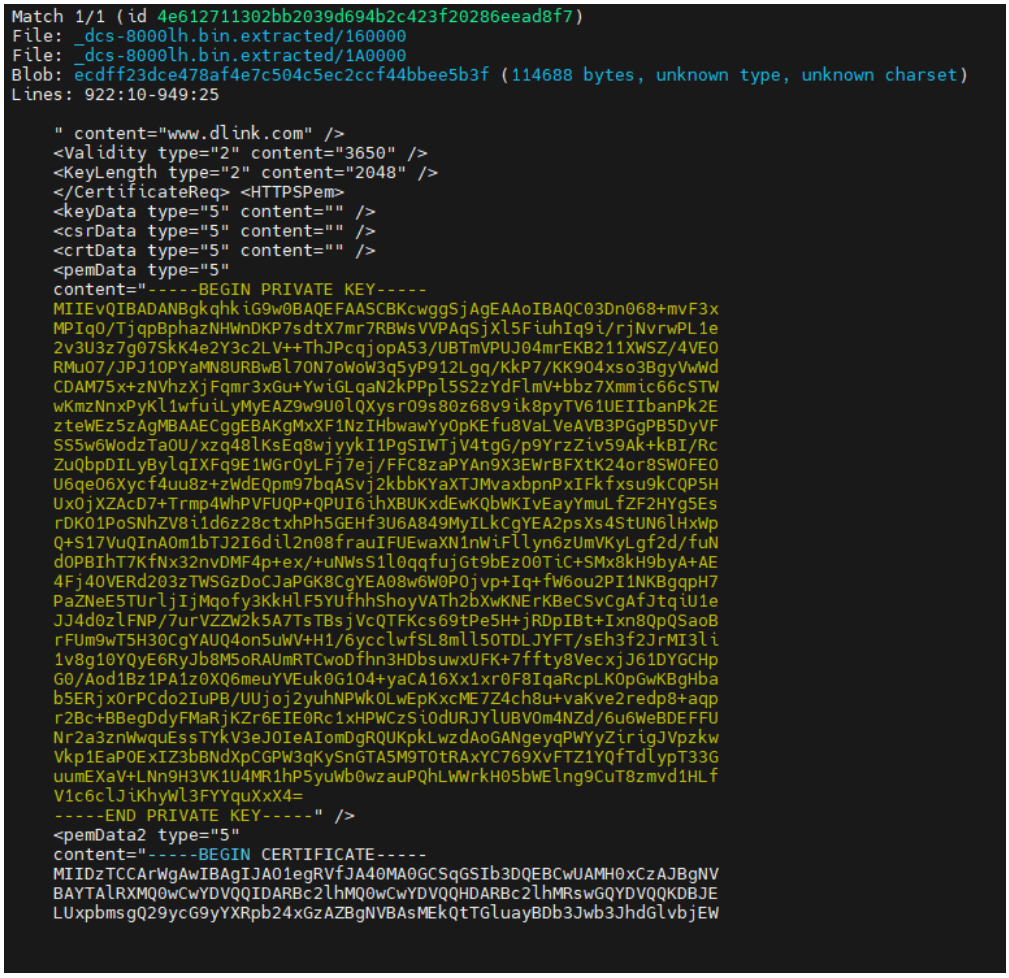


Figure 2.3



Analysis for bare-metal-demo.elf

-ELF 32-bit LSB executable, ARM, EABI5

- AWS access and secret key, GitHub Auth token, and personal access token, along with a PEM private key

Automation Script

In technology, establishing routine patterns and building in automation is almost always preferred. By doing so, it provides efficiency and also higher levels of accuracy, which is extremely important when trying to detect potential software vulnerabilities. That said, we developed a Python script to help in the analysis of .bin and .elf firmware images by automating the extraction and scanning of hardcoded secrets.

The script we built has two main capabilities. For .bin files, the script invokes binwalk, which is a firmware analysis tool that helps detect squashfs file systems that are usually stored on Linux IoT devices. That said, the script we developed looks to see if a

Squashfs file system is on the device; if it is, the location is stored for further review. Regarding the .elf file, we chose to use the software noseyparker, which was developed by Praetorian, to assist in the detection of stored secrets and sensitive data. Each time this script runs, noseyparker creates a temporary datastore to isolate the extracted data from the .elf file. This datastore is then used later to discover hardcoded secrets, such as passwords, tokens, and SSH keys.

Our method of execution for this script was in VS Code, which provides a user-friendly UI for executing scripts and viewing folders and files in a centralized location. This entire script runs in a few seconds and produces folders with extraction files for each .bin file and an overall findings.json file that contains the vulnerabilities it was able to discover. The vulnerabilities for the .elf file are also stored in the same .json file. As previously mentioned, the time savings and ability to view all vulnerability findings in a single location are the benefits of building out an automation script.

Figure 3.1