COSC 6840

Ethical Hacking

Midterm Project

ThanosTech LLC IoT Device Firmware Security Analysis

Andy Yothsackda, Jon Menzel, Alyssa Scott

TABLE OF CONTENTS

1. Introduction

1.1 Client Overview (Fictitious Description)

ThanosTech LLC, located in Milwaukee, WI, is an IoT device manufacturer that specializes in connected home and industrial monitoring technologies. The company has been in business for roughly ten years and has, unfortunately, been known to have vulnerability findings come up for its products. That said, their product department reached out to us (Andy, Alyssa, Jon) and asked if we could conduct a vulnerability scan on a few of their software images for their most popular devices. They are hoping to remediate future vulnerabilities so their brand reputation is not further impacted.

1.2 Assessment Scope

The agreement we signed consisted of performing analysis on the following firmware images (wr-841n.bin, dcs-8000lh.bin, and an ELF file). The analysis we agreed to conduct consisted of identifying potential vulnerabilities by running a software extraction script that searches for exposed credentials, passwords, and other easily identifiable vulnerabilities. Under this agreement, we have specifically stated that we would not be conducting any live device testing, exploitation, or code correction. This will just be a vulnerability scan/analysis. Below, we have specified in detail what our assessment will include.

The assessment included:

- Binary analysis of the firmware to understand the file system structure, kernel configuration, and embedded binaries.
- Detect vulnerability weaknesses by scanning for default settings, exposed services, and insecure implementations.
- Evaluation of identified weaknesses/vulnerabilities and their potential impact.
- Development of an automation script to streamline analysis and recommendations for changes in the name of device security.

The purpose of this contract was to strengthen ThanosTech LLC's security posture ahead of their deployment of their new line of products, guiding other future firmware development.

2. Methodology

2.1 Tools and Environment

- Tools used (binwalk, file, strings, grep, hexdump, readelf, Nosey Parker).
    - Python for language
    - binwalk- firmware extraction/analysis
    - File identification along with its format
    - Strings- extracting character sequence from its binary state
    - Grep- extraction of keyword searching such as “password” or “key” (not used but could be manually)
    - Hexdump- hex analysis (not used but could be manually)
    - Readelf- ELF architecture analysis
    - Nosey Parker: Does all of the above and additional scans. Automates and enhances all of the above by performing deep scans for hardcoded secrets, API keys, and private credentials within files, repositories, and extracted data.
        - Uses pattern-based detection and context analysis.
        - Detects sensitive items like AWS keys, GitHub tokens, private keys, and passwords.
        - Useful for security auditing, firmware recon, and data leakage detection.

2.2 Analysis Workflow

Workflow:

1. *Reconnaissance*
2. *Extraction*
3. *File Scanning*
4. *Vulnerability Discovery*
5. *Automation Scripting and Report*

We first started with the reconnaissance with tools such as binwalk, file, strings, grep, hexdump, readelf. Retrieving basic information such as filetype, U-Boot strings, LZMA compress kernels, SquashFS file system, Header signature patterns (using Wikipedia list of file signatures for classification). Then, after the lecture on 10/7, Nosey Parker was introduced to our class, which streamlined our tools and additional scans, and various types of deep scans that we weren't familiar with.

2.3 Firmware Characterization

During this stage, the goal was to understand the structure and security posture of each firmware file. The three firmwares were analyzed: wr-841n.bin, dcs-8000.bin, and bare-metal-demo.elf. The analysis combined basic file tools with Nosey Parker to identify sensitive or “juicy” information hidden inside.

wr-841n.bin

- U-Boot 1.1.3 bootloader at 0xD120
- LZMA-compressed Linux kernel at 0x10200 SquashFS root filesystem at 0x100000
- Metadata: MD5 97710bf8ae216d4665c1c89a43eca626, build date 2022-08-16

dcs-8000.bin

- Linux-based system with SquashFS
- Includes web interface files and startup/config scripts
- 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.

bare-metal-demo.elf

- Small ELF binary (~6.8 KB)

3. Findings

3.1 Vulnerability Overview

The firmware analysis found a few security issues, which are listed below:

Hardcoded Credentials: All three firmware files contained usernames and passwords written directly into the code. We found admin credentials, some encoded in base64 (which is relatively easy to decode using base64 decode online), and generic passwords like "admin/password."

Keys and Tokens: The firmware files have AWS access keys, GitHub tokens, and unencrypted private keys sitting in plain text. Anyone who extracts these firmware files could use these the default credentials and access their cloud accounts or private repositories to tamper with or take information.

Default Settings: Default IP addresses, network configurations, and device certificates were all exposed. The IP camera firmware even had Wi-Fi credentials and API keys embedded in configuration files. Each file had credential leaks, which the wr-841n.bin (router firmware) exposed admin passwords. The dcs-8000.bin (camera firmware) contained a full RSA private key. The bare-metal-takehome.elf file had AWS keys, GitHub tokens, and user credentials all hardcoded inside.

The vulnerabilities could let attackers access devices, access cloud services, and compromise/tamper with repositories. The firmware files show low security practices during the development processes. As well as passwords, keys, and other information pertaining to safety should not be hardcoded into firmware.

3.2 Security Impact Assessment

Overall, the firmware analysis showed a number of exposure of credentials, tokens, and private keys. These are weak security practices in firmware development, such as embedding secrets directly into code or the configuration files.

3.3 Evidence and Analysis

wr-841n.bin

- This firmware appeared to be for a network device. Binwalk, we identified the U-Boot 1.1.3 bootloader, an LZMA-compressed Linux kernel, and a SquashFS root filesystem. The extraction process worked but produced a warning about a missing sasquatch module. Metadata showed an MD5 hash of 97710bf8ae216d4665c1c89a43eca626 and a build date of August 16, 2022. Additionally, it revealed a complete root filesystem inside the configuration files and credentials. Noseyparker detected AWS keys, GitHub tokens, private keys, and even base64-encoded login information (admin:Password). These findings represent a critical security risk due to exposed credentials within the firmware.

1. Hardcoded Credentials- Base64-encoded YWRtaW4= (admin) / UGFzc3dvcnQ= (Password)
2. Generic username and password- admin and password (Figure 1.4, 1.5, and 1.6)
3. Default Configurations IP address is 192.168.15.15, U-Boot 1.1.3 bootloader, an LZMA-compressed Linux kernel, and a SquashFS root filesystem. MD5 hash of 97710bf8ae216d4665c1c89a43eca626 and a build date of August 16, 2022 (Figure 1.1 & 1.2)

   Evidence: See Initial Recon Attachment Figure 1.1 - 1.6

dcs-8000.bin

- This image came from an IP camera firmware. Inside, we found a Linux-based environment using SquashFS and web interface files. Configuration and startup scripts included admin credentials, Wi-Fi setup data, and device certificates. Nosey Parker scans confirmed the API keys and password strings embedded in the web configuration files.

- RSA Private Key- 2048-bit unencrypted PEM key (Figure 2.1, 2.2, & 2.3)

bare-metal-takehome.elf

- The bare-metal-takehome.elf file was small. Nosey Parker revealed several hardcoded secrets, including AWS credentials, GitHub tokens, and a PEM private key. Also user credentials were also found directly in the binary. This is a security issue, since anyone with the binary could obtain valid access keys and private information.

- AWS Secret Access Key, AWS API Credentials, GitHub Personal Access Token, PEM-Encoded Private Key, Generic Username and Password (Figure 3,1)

4. Automation Script

4.1 Script Purpose

The script automatically scans the firmware files to identify any juicy contents and/or any potential security issues. The script takes two types of firmware, .elf and .bin. The .elf file is scanned using noseyparker, which checks for vulnerabilities and code patterns. The .bin files are then extracted using Binwalk to scan for filesystems or data inside the firmware. The findings from both .elf and .bin files are combined and saved in a JSON report (report/findings.json). Extracted .bin files are stored in a separate folder (extracted/), which are ready for user inspection/ analysis or presentation to clients.

4.2 Execution Instructions

First, the virtual machine will need to have an updated version of noseyparker and Sasquatch to run the script. Upload the fw_triage.py and run python3 fw_triage.py on the command line. An output will confirm that the script ran successfully. An example is below.

- python3 fw_triage.py
    - Extracting firmware/dcs-8000lh.bin…
    - Extracting firmware/wr-841n.bin…
    - Running nosey parker…
    - All findings: report/findings.json
    - .Bin Saved to: extracted/

4.3 Time-Saving Benefits

This script saves time by automatically scanning and extracting multiple firmware files one at a time. It collects all results in one report, so it can avoid repetitive manual work. This allows them to focus on reviewing the findings instead of running scans.

4.4 Script Findings

The findings from an analysis of three firmware files:

1. dcs-8000lh.bin
2. wr-841n.bin
3. bare-metal-demo.elf

1. Firmware Extraction

- Dcs-8000lh.bin: Type: Binary (.bin)
    - Finding: Successfully extracted filesystem to extracted/extracted_dcs-8000lh.bin/_dcs-8000lh.bin.extracted/squashfs-root
- wr-841n.bin Type: Binary (.bin)
    - Finding: Successfully extracted filesystem to extracted/extracted_wr-841n.bin/_wr-841n.bin.extracted/squashfs-root

- bare-metal-demo.elf Type: Executable (.elf)
    - Finding: No filesystem extraction; this is a raw executable binary.

2. Security Findings in bare-metal-demo.elf

- AWS API Credentials, such as the access key ID and secret access key, were found in the binary. The AWS credentials would potentially allow attackers to access cloud information.
- GitHub Tokens- Access to the token allows access to GitHub repositories. Tokens could provide repository access, potentially allowing code tampering.
- Encoded private keys in PEM format are in the binary. Private keys can be used to authenticate or decrypt sensitive information..

3. Summary of Risks

- The bare-metal-demo.elf revealed secrets that are hardcoded in the binary, such as AWS credentials, GitHub access tokens, and PEM private key.
- The .bin firmware files don't contain secrets; however, the filesystems were extracted for further inspection then secrets were found afterwards.

5. Recommendations

Based on our analysis of the firmware and findings, we recommend the following changes:

5.1 Immediate Security Fixes

- Notify Customers
- Embedded Secret/Key Removal
    - Noseyparker scan identified several AWS access keys along with GitHub tokens, PEM encoded private keys, and base64 encoded login strings within the firmware. These credentials need to be revoked immediately, then new keys should be generated then stored using a secure configuration management system.
    - For any future firmware builds should never have access keys or credentials. Instead, implement runtime credential injection from secured storage. Integrate our new automated secret scanning tools, such as noseyparker, into the CI/CD pipeline to prevent future exposures.
- Disable Insecure Remote Access Services
    - Telnet and any web interfaces transmitting credentials in plaintext must be disabled. Replace the credentials within encrypted channels using modern cipher suites. Enforce key based multi factor authentication and with required password changes during first time setup to prevent unauthorized access.
- Update Outdated Core Components

    - The firmware relied on older versions should be updated to the latest vulnerability patched versions. Keeping a Software Bill of Materials for each firmware release will allow ThanosTech LLC to track component versions and respond quickly to future CVEs.

- Implement Firmware Integrity Verification
    - Enable cryptographic signing and secure boot to reassure only trusted firmware are loaded. This prevents malicious modification/tampering of the firmware during updates and deployments.

5.2 Future Enhancements To Strengthen Security Posture

- Integrate SonarQube for Code Quality Scans
    - Utilizing CI/CD pipelines, engineers will be able to build in a code quality scanner that will automatically search for software vulnerabilities. For example, if SonarQube was integrated prior to these images being productionalized, the quality scanner would have detected holes within the developed code. This would have been an instant red flag to the developers and would have prevented them from pushing it to production.
- Hire Ethical Hackers to Perform Vulnerability Scans
    - As an LLC that develops IoT devices, our strongest recommendation would be for you to hire ethical hackers who will perform rigorous testing on your devices and software images prior to them being deployed to production.
    - This will also be a positive for your brand reputation because it will provide your customers with a sense of security, knowing that your company is taking cyber threats seriously and that their data is protected.
- Conduct Threat Modeling Analysis
    - The vulnerabilities that were uncovered during this analysis were significant and show that there are gaps in your software development process. I would consult your cybersecurity team and software engineers, and get them aligned with the potential vulnerabilities. If these two orgs are not aligned, quality code will not exist.
    - STRIDE/PASTA Analysis
        - What type of threats exist?
        - How can this threat be mitigated, etc.
        - Perform risk analysis
- Require MFA/Security Keys to Access Devices
- Ensure All Ports Are Secured

6. Conclusion

In conclusion, we determined that ThanosTech LLC does have several security concerns that need to be addressed for their two most popular devices. Without addressing these concerns, their products are susceptible to potential attacks that may lead to a tarnished brand reputation. For example, we discovered through our analysis that hardcoded AWS credentials, GitHub Tokens, and

Private Keys existed in their .ELF file. Furthermore, .bin file, wr-841n.bin, contained a vulnerability where a network adjacent attacker would be able to access sensitive information, such as usernames and passwords that are currently stored on the device by listening on port 80. If this initial breach occurs, an attacker would be able to cause significant harm to the network and essentially control the entire device or monitor all activity on a network. This is a vulnerability that would put your customers at a heightened cybersecurity risk.

Lastly, we analyzed the .bin file, dcs-8000lh.bin, which contained a vulnerability that allows for the device's web interface to be exposed by accessing the following file location, <Camera-IP>/common/info.cgi. This location can be accessed without any form of authentication. Once this file has been accessed, the attacker would be able to get their hands on information that will allow them to have direct control over the device. For example, sensor settings, speaker, wireless connectivity status, etc. This again is a significant finding that not only places your brand at harm but also your customers. Our first recommendation is for these devices to be patched immediately, which will remove these vulnerabilities so that your devices are not exposed to outside threats. Overall, this project provided ThanosTech LLC with practical recommendations and tools to improve its security posture. The automation script we developed during the project offers an efficient and repeatable way to identify future vulnerabilities. By adopting these recommendations and making security a regular part of the development and testing process, ThanosTech LLC can continue to innovate with confidence while keeping user trust and data protection at the center of its mission.