Project Title: Firmware Reverse Engineering and Static Analysis

---

Short Project Description:

A system for analyzing firmware security through reverse engineering and static analysis. This system identifies vulnerabilities, extracts hardcoded credentials, compares firmware versions, and generates actionable reports.

Component	Role
Firmware Image Unpacker	Extracts filesystems, binaries from firmware
Static Analyzer Module	Scans code for vulnerabilities
Embedded Credentials Extractor	Searches for hardcoded secrets
Firmware Diffing Engine	Compares firmware versions for patch diffing
Reporting Engine	Summarizes firmware vulnerabilities and weaknesses

---

Component Details:

1. **Firmware Image Unpacker**:
    - Parses:
        - Compressed filesystems (squashfs, cramfs, etc)
        - Firmware headers
    - Extracts binaries and configs.
2. **Static Analyzer Module**:
    - Scans binaries for:
        - Unsafe API usage
        - Default credentials
        - Hardcoded keys
        - Etc
3. **Embedded Credentials Extractor**:
    - Greps for:
        - Passwords
        - API tokens
        - SSH keys
        - WiFi credentials
        - Etc
4. **Firmware Diffing Engine**:
    - Compares two firmware images:
        - Before and after updates
        - Find newly introduced or removed vulnerabilities
5. **Reporting Engine**:
    - Lists:

        - Weak credentials
        - Code vulnerabilities
        - Version comparisons
        - Etc

Overall System Flow:

- Input: Firmware binary
- Output: Vulnerability analysis report
- Focus: **Offline static firmware security assessment**.

Internal Functioning of Each Module:

1. Firmware Image Unpacker

- **Unpacking steps**:
    - **Binwalk**:
        - Identify embedded filesystems or binaries inside firmware.
        - Extract automatically if signatures match.
    - **Manual Extraction**:
        - For custom or obfuscated formats:
            - Entropy analysis.
            - Header reverse engineering.

---

2. Static Analyzer Module

- **Static scans**:
    - Analyze extracted binaries for:
        - Unsafe library usage (e.g., `strcpy`, `system`, `etc`).
        - Open administration interfaces (e.g., lighttpd configs with no password).
        - Outdated third-party components (by hashing binaries and matching known CVEs).

---

3. Embedded Credentials Extractor

- **Grep-based searches**:
    - Regex match for:
        - "admin"
        - "root"
        - Known password patterns
        - Etc
    - Search inside configuration files like `/etc/passwd`, `/etc/shadow`.

---

4. Firmware Diffing Engine

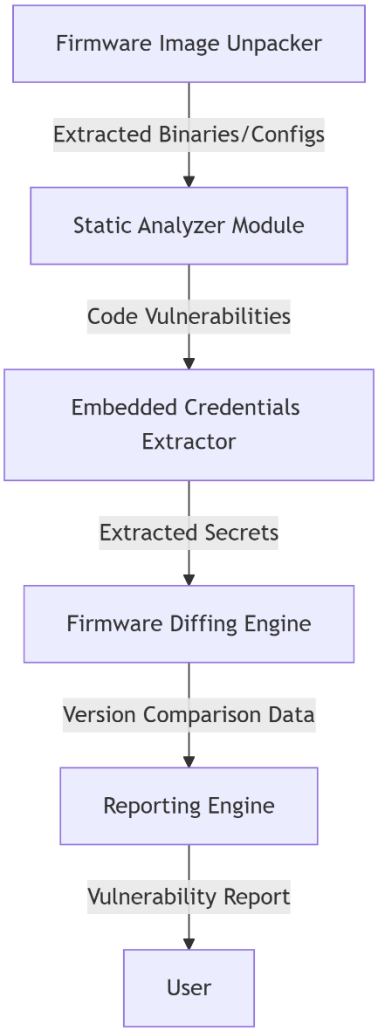
- **Diff Analysis**:
  - Compare firmware v1.0 vs v2.0:
    - New binaries
    - Patched binaries
    - Removed functionality
- **Delta Debugging**:
  - Highlight security-relevant changes (e.g., SSL updates, removed telnet servers).

---

5. Reporting Engine

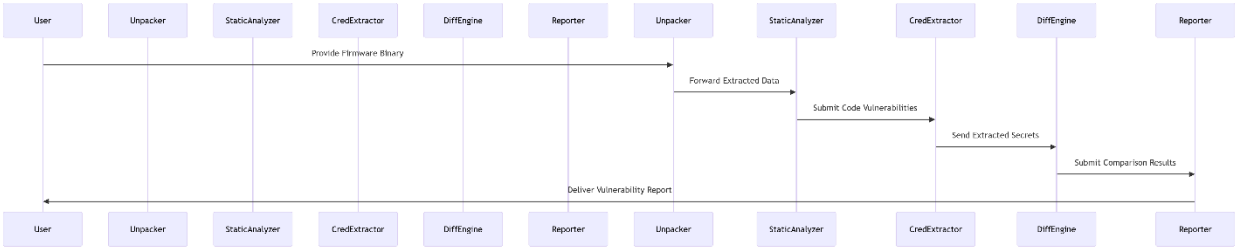
- **Comprehensive Reporting**:
  - List:
    - Weak credentials.
    - Vulnerable binaries.
    - Components that need patching.
    - Etc.
  - Risk scoring:
    - High (root shell available)
    - Medium (credentials extracted)
    - Low (outdated library with no known exploit)
    - Etc.

Component Diagram



- **Firmware Image Unpacker**: Extracts binaries, filesystems, and configurations from the firmware image.
- **Static Analyzer Module**: Scans extracted code for vulnerabilities (e.g., unsafe APIs, default credentials, etc).
- **Embedded Credentials Extractor**: Searches for hardcoded secrets (passwords, API keys, etc) using regex patterns.
- **Firmware Diffing Engine**: Compares firmware versions to identify patched or new vulnerabilities.
- **Reporting Engine**: Generates a report detailing vulnerabilities, credentials, and version differences.

Sequence Diagram



1. **User** provides a firmware binary to the **Firmware Image Unpacker**.
2. **Unpacker** extracts files and forwards them to the **Static Analyzer Module** for vulnerability scanning.
3. **Static Analyzer** submits detected code issues to the **Embedded Credentials Extractor** for secret extraction.
4. **Credentials Extractor** sends extracted secrets to the **Firmware Diffing Engine** for version comparison.
5. **Diffing Engine** analyzes differences (e.g., patched vulnerabilities) and sends results to the **Reporting Engine**.
6. **Reporting Engine** compiles findings into a final report for the **User**.

Detailed Project Description: Firmware Reverse Engineering and Static Analysis

A system for analyzing firmware security through reverse engineering and static analysis. This system identifies vulnerabilities, extracts hardcoded credentials, compares firmware versions, and generates actionable reports.

---

1. System Components and Roles

1.1 Firmware Image Unpacker

Purpose: Extract filesystems, binaries, and configurations from firmware images.
Implementation Details (e.g.):

- **Tools**:
  - **Binwalk**:
    ```
    binwalk -eM firmware.bin  # Extract embedded filesystems recursively
    ```
  - **Firmware Mod Kit (FMK)**: For manual extraction of custom/obfuscated formats.
  - **Etc**
- **Steps**:
  1. **Automated Extraction**: Use Binwalk to identify and extract common formats (squashfs, cramfs).
  2. **Manual Extraction**: For encrypted or custom formats:
     - **Entropy Analysis**: Detect encrypted regions using `binwalk -E firmware.bin`.
     - **Hex Editing**: Use `hexedit` or `dd` to carve out sections based on headers.
- **Example**:
  ```
  dd if=firmware.bin of=filesystem.squashfs bs=1 skip=123456  # Extract from offset
  unsquashfs filesystem.squashfs  # Decompress squashfs
  ```

1.2 Static Analyzer Module

Purpose: Scan extracted binaries for vulnerabilities and outdated components.
Implementation Details (e.g.):

- **Tools**:
  - **Ghidra/IDA Pro**: Decompile binaries to audit for unsafe APIs (e.g., `strcpy`, `system`).
  - **Checksec**: Check binary protections (NX, ASLR, Stack Canaries, etc).
  - **CVE Matching**: Hash binaries and query databases like NVD:

```
sha256sum binary.bin  # Get hash
curl https://services.nvd.nist.gov/rest/json/cves/1.0?cpeMatchString=cpe:2.3:a:vendor:binary:1.0
```

- **Checks**:
  - **Default Credentials**: Search for `admin:admin` in configuration files.
  - **Open Ports**: Analyze `inetd.conf` or `lighttpd.conf` for exposed services.
  - **Etc**

1.3 Embedded Credentials Extractor

Purpose: Identify hardcoded secrets (passwords, API keys).
Implementation Details (e.g.):

- **Tools**:
  - **grep**: Regex-based searches

```
grep -rE 'password\s*=\s*[\x22\x27].*[\x22\x27]' extracted_fs/  # Find passwords
grep -r 'BEGIN RSA PRIVATE KEY' extracted_fs/  # Find SSH keys
```

  - **TruffleHog**: Detect high-entropy strings (API tokens):

```
trufflehog --regex --entropy=6 filesystem/
```

- **Common Patterns**:
  - Default credentials (`root:root`, `admin:password`).
  - AWS keys (`AKIA[0-9A-Z]{16}`).
  - Etc

1.4 Firmware Diffing Engine

Purpose: Compare firmware versions to identify security patches or new vulnerabilities.
Implementation Details (e.g.):

- **Tools**:
  - **radiff2** (radare2): Binary diffing:
    ```
    radiff2 -AC firmware_v1.bin firmware_v2.bin  # Compare and highlight changes
    ```
  - **Binwalk Diff**:
    ```
    binwalk -W firmware_v1.bin firmware_v2.bin  # Compare entropy and signatures
    ```
- **Analysis**:
  - **Patch Detection**: Identify updated binaries (e.g., `libssl.so` with CVE fixes).
  - **Removed Services**: Check if insecure services (Telnet) were disabled.

1.5 Reporting Engine

Purpose: Generate structured reports summarizing vulnerabilities and recommendations.
Implementation Details (e.g.):

- **Tools**:
  - **Python + Pandas**: Aggregate results into CSV/JSON:
    ```
    import pandas as pd
    data = {"Vulnerability": "Hardcoded SSH Key", "Severity": "High"}
    df = pd.DataFrame([data])
    df.to_csv("report.csv")
    ```
  - **Jinja2 + LaTeX**: Generate PDF reports:
    ```
    \section{Vulnerabilities}
    \begin{itemize}
    \item \textbf{High}: Hardcoded root password in /etc/shadow.
    \end{itemize}
    ```
- **Metrics**:
  - **Risk Scores**: Critical (remote code execution), High (default credentials), Low (deprecated library).

---

2. System Integration and Component Interaction

1. **Unpacking**:
    - **Firmware Image Unpacker** extracts files → passes to **Static Analyzer**.
2. **Static Analysis**:
    - **Static Analyzer** flags vulnerabilities → sends to **Credentials Extractor**.
3. **Credential Extraction**:
    - **Credentials Extractor** finds secrets → forwards to **Diffing Engine**.
4. **Diffing**:
    - **Diffing Engine** compares firmware versions → sends results to **Reporting Engine**.
5. **Reporting**:
    - **Reporting Engine** compiles data into a PDF/HTML report.

---

3. Implementation Steps (e.g.)

3.1 Environment Setup

- **OS**: Ubuntu 22.04 LTS, etc.
- **Dependencies**:

```
sudo apt install binwalk git python3-pip radare2
pip install trufflehog pandas jinja2
```

3.2 Firmware Extraction

- **Automated Extraction**:

```
binwalk -eM firmware.bin
```

- **Manual Extraction** (if encrypted):

  Use **JTAG** or **SPI flash dumping** to extract raw firmware.

3.3 Static Analysis

- **Ghidra Workflow**:
    1. Import binary into Ghidra.

2. Analyze for unsafe functions (e.g., `strcpy`).
3. Export findings to CSV.

3.4 Credential Extraction

- **Regex Search**:
```
grep -rE 'pass(word|wd) *= *["'\'']?[^"'\'' ]+["'\'']?' extracted_fs/
```

3.5 Diffing Firmware Versions

- **Binary Diff**:
```
radiff2 -x firmware_v1.bin firmware_v2.bin > diff_results.txt
```

3.6 Report Generation

- **Jinja2 Template**:

  Run

```
<h1>Firmware Security Report</h1>
<ul>
{% for vuln in vulnerabilities %}
  <li>{{ vuln.severity }}: {{ vuln.description }}</li>
{% endfor %}
</ul>
```

---

4. Evaluation Criteria

1. **Detection Accuracy**: Percentage of known CVEs identified.
2. **False Positives**: Manual review of reported credentials/vulnerabilities.
3. **Diff Precision**: Correct identification of patched/unpatched issues.

---

5. Ethical and Legal Considerations

- **Authorization**: Only analyze firmware you own or have permission to test.
- **Compliance**: Follow responsible disclosure for discovered vulnerabilities.

---

6. Tools and Resources (e.g.)

- **Unpacking**: Binwalk, FMK, etc.
- **Analysis**: Ghidra, Checksec, etc.
- **Credentials**: TruffleHog, grep, etc.
- **Diffing**: radiff2, Binwalk, etc.
- **Reporting**: Pandas, LaTeX, etc.
