New issue

MikroTik RouterOS 6.49.8 Out-of-Bounds Read (nova/lib/www/scep.p) #4

Open

ezio315 opened last month · edited by ezio315 Edits Owner

Attachments

- 04-mikrotik-scep-p-cve-attachment.zip

Submitter

- Fengyi Wang
- Huazhong University of Science and Technology

1. Vulnerability Description

Within this submission package, bundled evidence files referenced below are stored under `./supporting/` and `./repro/`, relative to the directory containing this `report.md`.

An out-of-bounds read vulnerability exists in the `nova/lib/www/scep.p` component of **MikroTik RouterOS** firmware. The SCEP server parses attacker-controlled PKCS#7 signed attributes such as `transactionID` and `messageType` by returning raw `ASN1_STRING_data()` pointers and later consuming those values with NUL-terminated C-string semantics. An attacker who can reach an enabled SCEP endpoint can send a crafted `PKIOperation` request containing a non-NUL-terminated ASN.1 PrintableString and trigger reads beyond the original ASN.1 attribute boundary, leading to pre-authentication information disclosure in `certRep` replies and abnormal parser behavior.

This issue was analyzed against the following official firmware line:

- ARM build: `routeros-arm-6.49.8.npk`

- **Vulnerability Type**: Out-of-bounds Read (CWE-125), Improper Handling of Length Parameter Inconsistency (CWE-130)

2. Affected Product & Version

- **Vendor**: MikroTik
- **Product**: MikroTik RouterOS firmware
- **Verified Affected Version**:
  - `routeros-arm-6.49.8.npk` / `6.49.8` (ARM)
- **Affected Version Note**: At minimum, the verified RouterOS `arm` `6.49.8` package line is affected.
- **Firmware Download Links**:
  - `routeros-arm-6.49.8.npk`
  - `all_packages-arm-6.49.8.zip`
- **Vulnerable Component**: `nova/lib/www/scep.p`
- **Attack Surface**: `/scep/?operation=PKIOperation`

3. Root Cause Analysis

The root cause is loss of ASN.1 length information in the SCEP signed-attribute parsing path.

In the verified ARM sample, the relevant function chain is:

- `PKIMessage::getPString` at `0x96e8`
- `PKIMessage::decodeSignerInfo` at `0xb248`
- `PKIMessage::createReplyFrom` at `0x9be4`
- `PKIMessage::encodeSignerInfo` at `0xa468`
- `PKIMessage::addPString` at `0x13e20`

The vulnerable behavior can be summarized as:

```
char *s = PKIMessage::getPString(...);
msg_type = atoi(s);
transaction_id.assign(s);
...
ASN1_STRING_set(out_attr, transaction_id.c_str(), -1);
```

`getPString` returns `ASN1_STRING_data()` directly and does not preserve the original ASN.1 length. `decodeSignerInfo` then consumes `messageType`, `transactionID`, `pkiStatus`, and `failInfo` before signature verification. Later, the reply path copies `transactionID` from the request object and re-encodes it through `ASN1_STRING_set(..., -1)`, which derives the new length from `strlen()` rather than a trusted ASN.1 length.

If a malicious request supplies a length-valid PrintableString that is not NUL-terminated inside its intended ASN.1 boundary, both the parser and the reply-construction path may continue reading until a later NUL byte outside the original object. This is the precise condition for an out-of-bounds read.

4. Proof of Concept (PoC) & Steps to Reproduce

Verification was performed by generating a valid SCEP `PKIOperation`, mutating its signed attributes, and replaying the request to a live RouterOS SCEP endpoint.

Step 1: Generate a Valid Baseline SCEP Request

Enable SCEP server support and trigger a normal enrollment request. One verified RouterOS-side action is:

```
/certificate add-scep template=scep-clean-1 scep-url=http://127.0.0.1/scep/
```

This yields a valid CMS/PKCS#7 request body that can be extracted from RouterOS certificate debug logs. In the current public package, the included reproduction artifact is the ASN.1 parse used to locate the `transactionID` attribute:

- `./repro/chr_7.20.7_scep_client_pkio_asn1.txt`

Step 2: Mutate the Signed Attribute

Create a modified copy of the baseline request and alter the signed-attribute `transactionID`. A minimal demonstrated mutation changes the first byte:

- original: `cba0512e305210281e6dd2fac96699f080e6d14decdc3fca0abc815eb69486e0`
- mutated: `Zba0512e305210281e6dd2fac96699f080e6d14decdc3fca0abc815eb69486e0`

The demonstrated mutation can be reproduced by editing the `transactionID` field in a valid baseline request body using the ASN.1 offsets described in the included parse output.

Step 3: Replay the Request

Replay the mutated request to the SCEP endpoint:

```
curl -sS -X POST "http://127.0.0.1/scep/?operation=PKIOperation" \
  -H "Content-Type: application/x-pki-message" \
  --data-binary @./mutated-request.der
```

Step 4: Observation

- The server parses the attacker-controlled `transactionID` value before signature verification fails.
- A clean baseline request shows that the same `transactionID` is later reused in both `certRep pending` and `certRep success`.
- This demonstrates that attacker-controlled signed attributes reach the live reply-construction path.

The same replay workflow can be used with a non-NUL-terminated PrintableString payload to exercise the over-read condition directly.

5. Debugging Evidence

5.1 Static ARM Sample (`routeros-arm-6.49.8.npk`)

Static analysis confirms the vulnerable chain.

- `PKIMessage::getPString` returns raw `ASN1_STRING_data()` for ASN.1 PrintableString-type attributes.
- `PKIMessage::decodeSignerInfo` consumes `messageType` with `atoi()` and `transactionID` with `string::assign()` before signature verification.
- `PKIMessage::createReplyFrom` copies the request `transactionID` into the reply object.
- `PKIMessage::encodeSignerInfo` and `PKIMessage::addPString` serialize reply attributes through `ASN1_STRING_set(..., -1)`.

Together, these observations show that the parser loses explicit length information and then reuses attacker-controlled data with C-string semantics.

5.2 Live Parsing Evidence

Runtime SCEP logging shows that signed attributes are consumed before trust is established.

For the mutated request:

- original `transactionID`: `cba0512e305210281e6dd2fac96699f080e6d14decdc3fca0abc815eb69486e0`
- mutated `transactionID`: `Zba0512e305210281e6dd2fac96699f080e6d14decdc3fca0abc815eb69486e0`

Observed sequence:

- RouterOS logs the mutated signed-attribute value first.
- Only afterwards does the server emit:
  - `signed attribute signature not matching`
  - `signature verify failed`

This confirms that an unauthenticated attacker can reach the vulnerable parsing path before signature verification blocks the request.

5.3 Live Reply-Path Evidence

Runtime validation also confirms that the live SCEP server reuses request `transactionID` values in replies.

The clean sample was:

- `common-name`: `scep-clean-1`
- `transactionID`: `2153e85a90763000b53560a9d9ff5153c76ac16d0445f8c1886f8e404fd7f210`

- issued certificate serial: `584FDBDDAB6DCFA5`

Observed sequence:

- At `15:08:51`, the server parses the request and returns `certRep (3)` with `status: pending (3)`.
- At `15:08:53`, the server logs:
  - `granted certificate enrollment transaction 2153e85a90763000b53560a9d9ff5153c76ac16d0445f8c1886f8e404fd7f210`
- At `15:09:21`, the client polls `getCertInitial (20)` using the same `transactionID`.
- At `15:09:22`, the server logs:
  - `generated certificate 584FDBDDAB6DCFA5:scep-clean-1`
  - `packet encoding message type: certRep (3)`
  - `packet transaction: 2153e85a90763000b53560a9d9ff5153c76ac16d0445f8c1886f8e404fd7f210`
  - `scep-clean-1 certificate updated`

Offline inspection of the logged success `certRep` confirms that the success reply body contains:

- `messageType=certRep (3)`
- `pkiStatus=success (0)`
- `senderNonce`
- `recipientNonce`
- `transactionID`

The ASCII-encoded `transactionID`

- `2153e85a90763000b53560a9d9ff5153c76ac16d0445f8c1886f8e404fd7f210`

is present inside the success reply body. This proves that the live reply path really re-encodes request `transactionID` values.

Supporting package files are included under:

- `./supporting/static_root_cause.md`
- `./supporting/response_path_verified.md`
- `./repro/`

6. Security Impact

This vulnerability allows a remote unauthenticated attacker who can reach an enabled SCEP endpoint to:

- trigger out-of-bounds reads while RouterOS parses attacker-controlled signed attributes
- disclose unintended memory bytes through `certRep` reply construction
- cause abnormal parser behavior or malformed reply generation

Because the vulnerable parsing occurs before signature verification, exploitation does not require a pre-trusted enrollment identity. The practical impact is best characterized as **pre-authentication information disclosure**.

7. Suggested Mitigation

- Preserve explicit ASN.1 lengths for all signed-attribute consumers in `scep.p`.
- Replace `atoi()` and raw `string::assign(char *)` style handling with length-aware parsing.
- Pass explicit trusted lengths into `ASN1_STRING_set` instead of relying on `-1` / `strlen`.
- Reject malformed or non-canonical PrintableString values before reply construction.
- If possible, delay consumption of security-relevant signed attributes until after signature validation.
- Audit sibling SCEP parsing paths for the same length-confusion pattern.

Metadata

Assignees

No one assigned

Labels

No labels

Projects

No projects

Milestone

No milestone

Relationships

None yet

Development

No branches or pull requests

Participants