



BGD e-GOV CIRT

TLP: CLEAR



CYBER THREAT ADVISORY

**Node2Shell (CVE-2026-111497)
Critical Brotli Decoder
Vulnerability Under Active
Exploitation**

**TLP:** CLEAR**Distribution:** Public**Advisory on:** Node2Shell (CVE-2026-111497): Critical Brotli Decoder Vulnerability Under Active Exploitation**Severity:** Critical**Vulnerability Type:** Out-of-Bounds Write / Memory Corruption**CVE IDs:** CVE-2026-111497**CVSS v4.0:** 9.2**Affected Component:** Brotli decompression library and products incorporating vulnerable implementations**Date:** 29 October 2026

Executive Summary

BGD e-GOV CIRT alerts government agencies, critical infrastructure operators, financial institutions, telecommunications providers, software developers, hosting providers and other organizations to a critical vulnerability in the Brotli decompression library, tracked as **CVE-2026-111497**.

The vulnerability, commonly referred to as **Node2Shell**, is an out-of-bounds write in the Brotli decoder. A carefully malformed Brotli-compressed data stream can trigger an invalid memory write during decompression. Depending on the affected application and runtime environment, successful exploitation may result in process crashes, disclosure of process memory or remote code execution (RCE).

The vulnerability affects Node.js applications that use the vulnerable decoder and may also affect other software incorporating vulnerable Brotli implementations. Applications can be exposed when they decompress untrusted data through direct library calls, HTTP response processing or request-body parsing.

Public threat intelligence reporting indicates that exploitation is occurring in the wild. **The vulnerability was also reported as added to the Known Exploited Vulnerabilities (KEV) catalogue on 27 October 2026.** Organizations should therefore prioritize asset discovery, vulnerability assessment and deployment of the appropriate security updates.

The vulnerability is not restricted to servers. Applications that retrieve and automatically decompress untrusted HTTP responses may also be at risk. Exposure depends on the vulnerable component, the installed version, the data-processing path and whether attacker-controlled Brotli data can reach the decoder.

Immediate action: Identify vulnerable Node.js and Brotli deployments, upgrade to verified fixed releases or patched vendor packages, review other products incorporating Brotli, and investigate suspicious activity involving affected applications.

Vulnerability Overview

Brotli is a lossless compression algorithm used to reduce the size of data transmitted over networks and stored by applications. It is commonly associated with HTTP content encoding through the br encoding token.

CVE-2026-111497 affects the Brotli decoder's handling of malformed block-switch information in a compressed stream. The decoder may incorrectly interpret an invalid condition as a temporary shortage of input instead of rejecting the malformed stream.

As a result, the streaming decoder can continue appending data beyond the intended boundary of an internal buffer. This out-of-bounds write can corrupt adjacent process memory.

The vulnerability has been described under several names:

- **Node2Shell:** Commonly used in reporting focused on Node.js.
- **Broccoli2Shell:** Used in reporting about the broader Brotli vulnerability.
- **HTTP2Shell:** Another name appearing in public reporting.

These names refer to the same CVE in the sources reviewed; they should not be treated as separate vulnerabilities without additional evidence.

Vulnerability details

Attribute	Details
CVE	CVE-2026-111497
Component	Brotli decoder
Vulnerability class	Out-of-bounds write (CWE-787)
Reported CVSS	9.2, Critical
Primary trigger	Malformed Brotli-compressed input
Authentication	Not required for the vulnerable processing path described in the research
Potential impact	Memory corruption, process crash, information disclosure or RCE
Exploitation status	Active exploitation reported by public threat intelligence
Complete remediation	Apply a verified security update for the affected component

The reported CVSS score should be interpreted alongside the applicable product advisory. The practical risk depends on whether the affected application processes untrusted Brotli data and how the application is deployed.

Technical Analysis

Root cause

The vulnerability occurs in the Brotli decoder's block-type switch handling.

During decompression, the decoder processes a structured stream containing information about compressed data blocks. A malformed stream can signal an invalid block-type transition.

The vulnerable decoder incorrectly treats the invalid condition as though additional input might resolve it. In the streaming path, further bytes are appended to an internal buffer that has a fixed boundary.

The decoder subsequently writes beyond the intended buffer boundary, potentially corrupting memory belonging to the running process.

The technical issue can be summarized as:

- The application receives attacker-controlled Brotli-compressed data.
- The data reaches the vulnerable decoder.
- The stream contains an invalid block-switch condition.
- The decoder fails to reject the malformed condition correctly.
- Processing continues and causes an out-of-bounds write.
- The affected process may crash or experience memory corruption, with possible information disclosure or code execution.

This sequence describes the reported vulnerability mechanism. It does not mean that every malformed stream will produce reliable code execution.

Why the decoder's error handling matters

A decompression implementation must distinguish between incomplete input and structurally invalid input.

An incomplete stream may legitimately require more bytes before decoding can continue. A malformed stream, however, must be rejected when its structure is invalid.

In the vulnerable implementation, a particular invalid block-switch condition is treated as a request for more input. This allows the streaming decoder to continue processing when it should instead return a format error.

The upstream correction makes the decoder reject the invalid block-switch condition rather than treating it as incomplete input.

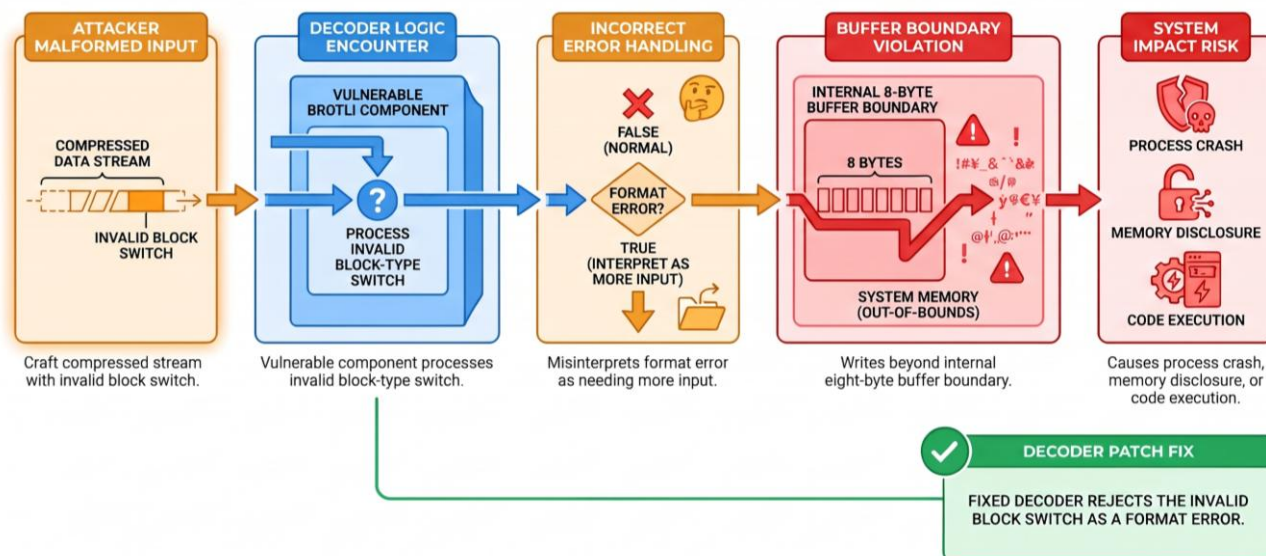


Fig: Conceptual sequence showing how invalid Brotli block-switch handling can cause an out-of-bounds write in vulnerable decoder implementations, potentially resulting in memory corruption, denial of service, information disclosure, or code execution.

Potential security impact

Successful exploitation may lead to:

- Denial of service: The affected application or worker may crash.
- Memory disclosure: Information residing in process memory may become exposed under exploitable conditions.
- Remote code execution: Memory corruption may potentially be leveraged to execute unauthorized code in the context of the affected process.
- Application compromise: If execution is achieved, an attacker may attempt to access application data, credentials, files or connected services available to that process.
- Lateral movement: A compromised service may become a starting point for further activity if network access or credentials permit it.

The exact outcome depends on the application, memory layout, runtime hardening and the attacker's ability to exploit the corruption. A proof of memory corruption or a process crash is not, by itself, proof of reliable remote code execution.

Affected Environments

The vulnerability may affect applications and systems that incorporate a vulnerable Brotli decoder and process attacker-controlled compressed data. Potentially relevant environments include:

- Node.js applications using the built-in `node:zlib` module.
- Applications using `zlib.brotliDecompress()` or `zlib.createBrotliDecompress()` on untrusted input.
- Applications that automatically decompress Brotli-encoded HTTP responses through the built-in `fetch()` API.
- Express 5 applications and applications using body-parser 2 where supported request parsers accept compressed request bodies.
- Browsers and other software embedding a vulnerable Brotli implementation.
- Web servers, reverse proxies, content-processing services and other software that decompresses untrusted Brotli data.
- Containers, application images and bundled runtimes that include vulnerable components.
- Applications using a standalone Brotli library or a downstream package containing the vulnerable implementation.

Important: Installing Node.js is not sufficient by itself to establish exposure. Administrators should determine whether the deployed application includes the affected implementation and whether untrusted compressed data can reach it.

Similarly, a web server does not necessarily decompress incoming Brotli request bodies automatically. The behavior depends on the web server, proxy, application framework and middleware configuration.

Affected and Fixed Node.js Versions

The published technical analysis identifies the following affected Node.js release lines.

Node.js release line	Affected versions	Fixed version
22.x	Versions before 22.23.4	22.23.4
24.x	Versions before 24.22.1	24.22.1

26.x	Versions before 26.12.1	26.12.1
25.x, 23.x, 20.x and earlier discontinued lines	Treat as affected where the vulnerable component is present	Migrate to a supported, patched release line

Organizations should verify the exact installed build and package source before considering an asset remediated.

Operating-system distributions may supply security fixes through their own packages, and the package version string may not match the upstream version number. Administrators should consult the relevant distribution security notice and confirm that the installed package contains the fix.

Identify deployed Node.js versions

On Linux or Unix-like systems, administrators can check the runtime version with:

```
node --version
```

Where applicable, also check:

```
nodejs --version
```

```
npm --version
```

On Windows, run:

```
node --version
```

```
npm --version
```

These commands identify the runtime available through the current environment; they do not necessarily enumerate every Node.js binary, container, bundled runtime or service-specific installation.

Organizations should also inspect:

- Running Node.js processes and executable paths.
- System package inventories.
- Container images and deployed containers.
- Application deployment manifests and lockfiles.
- CI/CD build images.
- Serverless runtimes and managed application platforms.
- Applications that bundle or statically link a Brotli implementation.

Exposure Paths and Potential Attack Scenarios

Direct decompression of untrusted input

An application may pass user-supplied or externally retrieved compressed data to a Brotli decompression API.

If the input reaches a vulnerable decoder, malformed compressed data may trigger the memory-corruption condition.

Examples include applications processing uploaded compressed files, externally supplied data streams or content retrieved from untrusted sources.

HTTP response decompression

Applications using HTTP clients may automatically decompress Brotli-encoded responses.

For example, a Node.js application using `fetch()` may receive a Brotli-encoded response from an external service and process it through the runtime's decompression implementation.

If an attacker controls the responding endpoint or can compromise an upstream service, maliciously formed compressed data may reach the decoder.

The affected component may be the client-side application rather than an Internet-facing server.

Compressed request bodies

Certain application frameworks and middleware support decompression of incoming request bodies.

The cited technical research identifies Express 5 and body-parser 2 parsers as relevant exposure paths when compressed request bodies are accepted and the parser's inflate setting is enabled.

In these circumstances, a remote client may be able to submit a Brotli-encoded body that the application attempts to decompress.

A bare Node.js HTTP server does not necessarily decompress Brotli-encoded request bodies on its own. The application's request-processing logic must be assessed to determine whether the vulnerable decoder is reached.

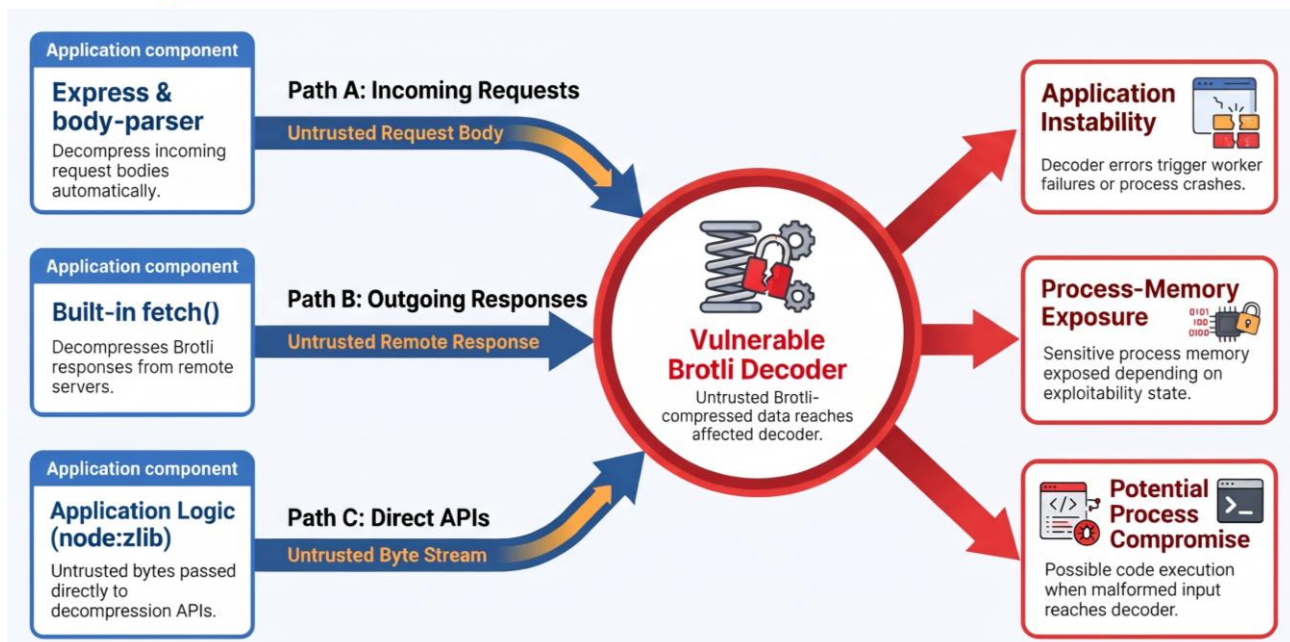


Fig: Node2Shell Attack Surface: Three Paths for Attacker-Controlled Brotli Data to Reach the Decoder

Broader Brotli Ecosystem Risk

The official security notice describes temporary settings that may reduce exposure to specific paths associated with these vulnerabilities. Administrators should apply them only after assessing application compatibility and operational risk.

Although Node2Shell received significant attention through Node.js, the underlying issue is in the Brotli decoder implementation. Other products embedding or distributing a vulnerable version may also require updates.

Organizations should not limit their response to Node.js servers. Review the following categories:

- Browsers and desktop applications that incorporate Brotli.
- Standalone Brotli libraries installed through operating-system packages.
- Applications that bundle native libraries.
- Web servers, proxies and content-processing software that decompress untrusted data.
- Container base images and application dependencies.
- Managed services for which the underlying runtime is maintained by a service provider.

The affected-product list and remediation process can differ between products. A fix to Node.js does not automatically fix a separate standalone Brotli package or an embedded copy in another application.

Likewise, a vulnerability announcement involving Brotli does not prove that every product using Brotli is exploitable. Product-specific advisories and deployment details must be checked

Immediate Mitigation and Remediation

Upgrade affected Node.js deployments

Upgrade affected Node.js installations to the appropriate fixed release:

- Node.js 22.x: 22.23.4 or later within the applicable supported release line.
- Node.js 24.x: 24.22.1 or later within the applicable supported release line.
- Node.js 26.x: 26.12.1 or later within the applicable supported release line.

Organizations running discontinued release lines should migrate to a supported release line that includes the security fix. After updating:

- Restart the affected services.
- Verify the running runtime version.
- Confirm the deployed application is using the updated executable or container image.
- Review service logs for errors introduced by the update.
- Retest critical request and response processing workflows.

Apply product-specific Brotli updates

For standalone libraries, browsers, web servers, proxies and other applications, install the security update supplied for the relevant product.

Do not assume that updating Node.js also patches every other Brotli implementation installed on the same system.

Temporarily restrict compressed request bodies

Where an application does not require compressed request bodies, administrators using the relevant body-parser configuration can disable request-body inflation:

```
inflate: false
```

For the applicable parser, compressed request bodies will then be rejected rather than decompressed. This setting is an exposure-reduction measure for that request-processing path. It does not mitigate every possible route to a vulnerable Brotli decoder, such as automatic decompression of HTTP responses or direct decompression-library calls.

Test the change before deployment because legitimate clients may rely on compressed request bodies.

Restrict network access and segment services

Where patching cannot be completed immediately:

- Restrict access to affected services to required sources.
- Reduce unnecessary Internet exposure.
- Limit outbound network access from affected application servers.
- Separate vulnerable workloads from sensitive internal systems.
- Restrict service-account permissions and access to secrets.
- Monitor attempts to reach affected services from unexpected sources.

Network restrictions can reduce exposure and limit the impact of compromise, but they do not replace a security update.

Rebuild and redeploy containers

For containerized applications:

- Update the base image or package containing the vulnerable component.
- Rebuild the application image.
- Scan the resulting image and dependencies.
- Redeploy the patched image.
- Confirm that old vulnerable containers and images are no longer serving production traffic.

Updating a package on the host does not necessarily update a separately bundled library inside a container.

Detection and Threat Hunting

Application and network telemetry

Review:

- Requests containing Content-Encoding: br.
- Endpoints accepting compressed request bodies.
- Unusual request sizes or repeated decompression errors.
- Unexpected HTTP response-processing failures.
- Application crashes and abnormal worker restarts.
- Suspicious requests to upload, import, parsing or content-processing endpoints.
- Unexpected outbound connections from Node.js services.
- Unusual interactions with external services that provide compressed responses.

The presence of Content-Encoding: br is normal in many environments. It should be correlated with application behavior and other indicators rather than treated as inherently malicious.

Endpoint and runtime monitoring

Investigate:

- Unexpected child processes spawned by Node.js services.
- Node.js processes launching shells or unrelated executables.
- Unexpected changes to application files or runtime configuration.
- New files in temporary or application directories.
- Unusual access to environment variables, credentials or service tokens.
- Unexplained outbound connections.
- Memory-related crashes or repeated abnormal process termination.

These are general indicators of possible exploitation or post-exploitation, not confirmed CVE-specific signatures.

SIEM correlation model

Prioritize investigation when several of the following occur in the same application context:

- Suspicious compressed input or an anomalous decompression error.
- Unexpected process crash or abnormal memory behavior.
- Unexpected child-process execution.
- An unusual outbound connection.
- Unauthorized file changes or access to sensitive credentials.

Correlate these events with the affected runtime version and whether the application can process attacker-controlled Brotli data.

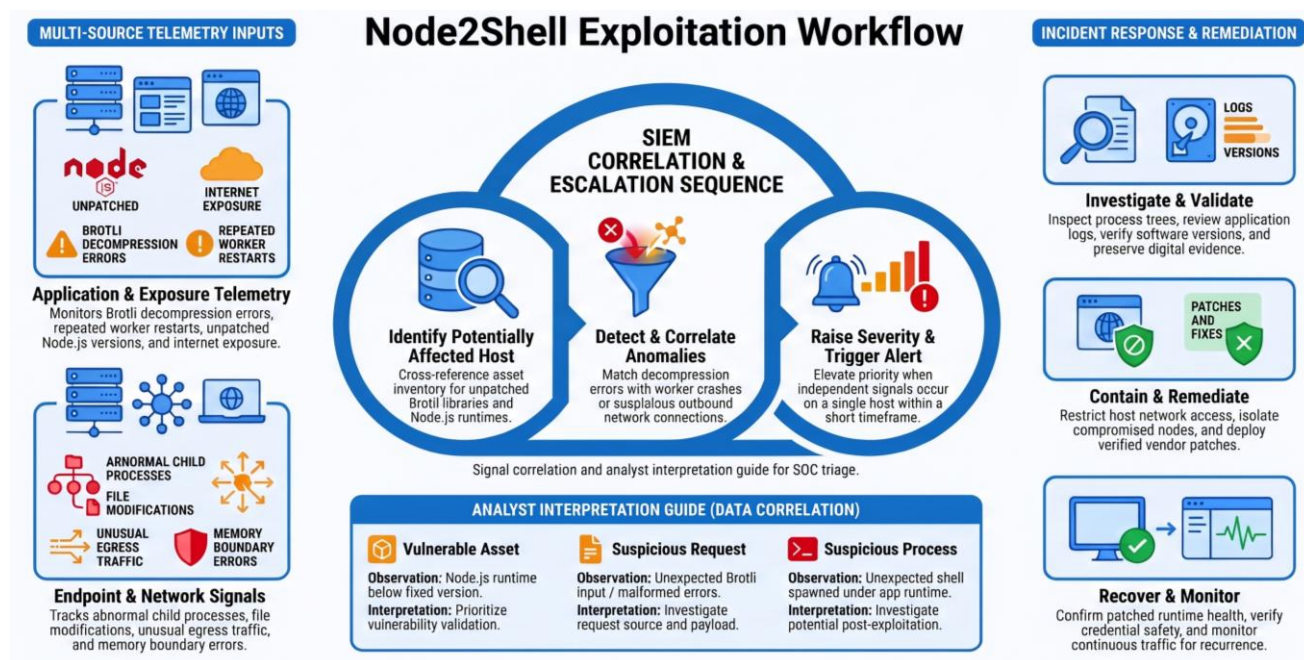


Fig: Detection and Response Workflow for Suspected Node2Shell Exploitation

Incident Response

If a system running a potentially vulnerable component exhibits suspicious activity:

- **Validate exposure:** Confirm the affected component, installed version and data-processing paths.
- **Contain:** Restrict access or isolate the affected service where operationally feasible.
- **Preserve evidence:** Collect application, web-server, process, system and network logs.
- **Investigate:** Review compressed-input handling, request patterns, process trees, file changes and outbound communications.
- **Protect credentials:** Rotate potentially exposed credentials, API keys and service tokens if compromise is suspected.
- **Eradicate:** Remove unauthorized files or processes and rebuild the service if integrity cannot be established.
- **Patch:** Install the appropriate security update and rebuild affected images or deployments.
- **Recover:** Validate application behavior and restore service from trusted components.
- **Monitor:** Increase monitoring for repeated exploitation attempts and potential lateral movement.

Do not assume that patching eliminates an attacker's existing access or proves that the affected system was never compromised.

Recommendations

BGD e-GOV CIRT recommends that government agencies, financial institutions, telecommunications providers, critical infrastructure operators, educational institutions, hosting providers and software development teams take the following actions.

Asset management

- Inventory Node.js runtimes, standalone Brotli packages and embedded libraries.
- Identify public-facing services and applications processing externally supplied compressed data.
- Include containers, build pipelines, application images and managed platforms in the inventory.
- Identify unsupported runtimes and prioritize migration.

Patch management

- Prioritize affected, Internet-accessible systems because active exploitation has been reported.
- Apply verified fixes for each affected component.
- Confirm the running service uses the patched build.
- Rebuild and redeploy containers when required.
- Track remediation for applications maintained by third-party providers.

Network and application security

- Restrict access to affected services where possible.
- Disable compressed request bodies where not required and supported by the application.
- Review request parsing and decompression behavior.
- Limit unnecessary outbound connectivity from application servers.
- Enforce least privilege and restrict access to sensitive internal systems.

Security operations

- Add affected software versions to vulnerability-management workflows.
- Review logs for anomalous decompression behavior and process activity.
- Correlate suspicious input with endpoint and network telemetry.
- Investigate credible indicators of exploitation.
- Share validated findings with relevant security stakeholders through established reporting channels.

Third-party and managed services

- Where organizations do not directly administer the runtime, they should request confirmation from the service provider regarding:
- Whether the affected component is present.
- Whether a security fix has been applied.
- Whether all production instances have been updated.
- Whether incident monitoring has been performed where appropriate.
- Whether any further action is required from the customer.

CVE-2026-111497 is a critical memory-corruption vulnerability in the Brotli decoder that can affect Node.js applications and other software incorporating vulnerable Brotli implementations.

The vulnerability is especially concerning because attacker-controlled compressed data may reach the decoder through direct decompression APIs, request-body parsers or automatic HTTP response processing. Public reporting indicates active exploitation, making timely remediation a priority.

Organizations should identify affected components, deploy the appropriate security updates, review exposure paths, and investigate suspicious activity. Temporary controls can reduce risk, but they do not replace patching.

BGD e-GOV CIRT urges all organizations to prioritize the assessment and remediation of CVE-2026-111497 across Node.js, Brotli libraries, containers and other affected software components.

References

- <https://www.sophos.com/en-us/blog/node2shell-cve-2026-111497-in-active-exploitation>
- <https://www.akamai.com/blog/security-research/inside-node2shell-closer-look-cve-2026-111497>
- <https://www.esri.com/arcgis-blog/products/trust-arcgis/administration/brotli-vulnerabilities-broccoli2shell>
- <https://access.redhat.com/security/vulnerabilities/RHSB-2026-015>



BGD e-GOV CIRT