GGSec Cortex · GG Advanced IT Security

Security Assessment Report

GGSec Cortex v0.8a Alpha · AI-Guided DAST · Context-Aware SAST · Target: template_engine.php · Generated: 2026-07-01 16:36

CONFIDENTIAL

Executive Summary

GGSec Cortex identified 2 confirmed and 7 likely vulnerabilities. Combined exploitation enables an unauthenticated attacker to achieve remote code execution. Every confirmed finding is evidence-backed (live proof-of-concept or a baseline-difference control), so false positives are minimised. **Immediate remediation is recommended.**

Template Engine API with two critical vulnerabilities: a second-order SSTI (stored template body is later eval'd via {{…}} pattern matching), and a blind NoSQL $where injection via the 'filter' JSON field. Both require authentication (session cookie).

Security Rating	**F · CRITICAL (2 Critical, 4 High)**
Max CVSS (v3.1)	**9,8 (Critical)**
Severity distribution	Critical: 4 High: 5 Medium: 0 Low: 0
Compromise probability	Very High (~95-100%)
Confirmed findings	**2**
Likely (needs review)	**7**
Business impact	**Full server compromise (remote code execution)**
Exploitation complexity	Low — reachable by a remote attacker
Likelihood	High — confirmed with a live proof-of-concept
Scan	SAST 0s · DAST 12s · 12 requests

Assessment Scope

Target	template_engine.php
Base URL	http://localhost
Endpoints tested	4
Total requests	12
HTTP methods	GET, POST, SCA
Vulnerability classes	3
Authentication	Unauthenticated
Scan duration	SAST 0s · DAST 12s
Completed	2026-07-01 16:36

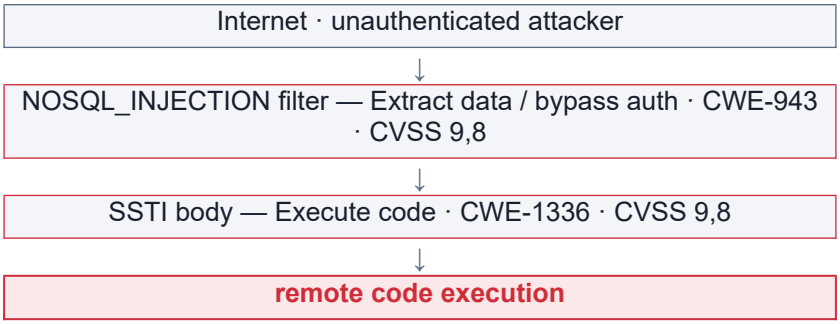
Scan Timeline

16:36:17 ● **Scan started**
16:36:17 ○ SAST replay (cached plan)
16:36:17 ○ DAST probing started
16:36:17 ○ GGC-SSTI-796 confirmed — SSTI body
16:36:21 ○ GGC-NOSQL-089 confirmed — NOSQL_INJECTION filter
16:36:21 ● **Attack chain verified → remote code execution**
16:36:29 ○ DAST completed (12s, 12 requests)

16:36:30 • **Report generated**

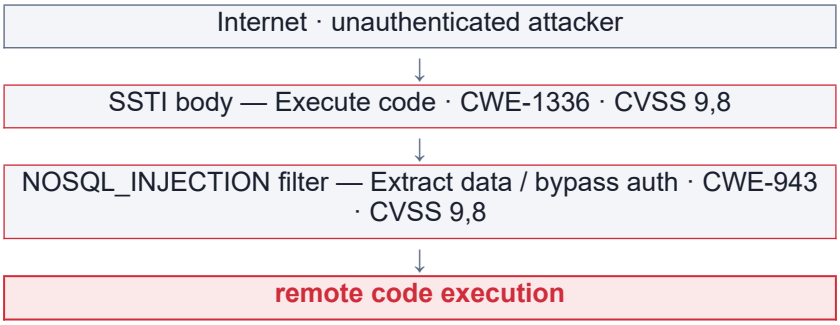
Exploitation Paths

Goal: remote code execution · Chain severity 10,0 (worst single 9,8)



Unauthenticated attacker → 1. extract data / bypass auth via NOSQL_INJECTION on 'filter' (CWE-943) → 2. execute code via SSTI on 'body' (CWE-1336) □ remote code execution.

Goal: remote code execution · Chain severity 10,0 (worst single 9,8)



Unauthenticated attacker → 1. execute code via SSTI on 'body' (CWE-1336) → 2. extract data / bypass auth via NOSQL_INJECTION on 'filter' (CWE-943) □ remote code execution.

Risk Matrix

Likelihood \ Impact	Negligible	Minor	Moderate	Major	Severe
Almost certain					1
Likely					
Possible					1
Unlikely					
Rare					

Endpoint Summary

Endpoint	Requests	Findings	Risk
`template_engine.php`	4	OUTDATED_COMPONENT, SSTI	Critical
`http://localhost/template_engine.php/save`	3	NOSQL_INJECTION	Critical
`tests\package-lock.json`	3	OUTDATED_COMPONENT	Critical
`tests\composer.lock`	2	OUTDATED_COMPONENT	High

Remediation Plan

ID	Finding	CVSS	Priority	SLA

GGC-NOSQL-089	NOSQL_INJECTION — filter	9,8	P0	Fix immediately
GGC-SSTI-796	SSTI — body	9,8	P0	Fix immediately

Findings (9)

[1] GGC-NOSQL-089 · NOSQL_INJECTION — filter

CONFIRMED CVSS 9,8 Critical · CWE-943 Improper Neutralization of Data in a Query (NoSQL) · Priority P0 (Fix immediately)

CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

Location: POST /save (JSON body field 'filter')

Impact: RCE (Direct)

Flow: The 'filter' field from the JSON POST body is decoded via json_decode into an associative array. If the resulting array contains a '$where' key (i.e. the attacker sends {"filter":{"$where":"..."}}) the code enters the sleep(4) branch, simulating server-side JavaScript evaluation in MongoDB's $where operator. In a real MongoDB deployment this would allow arbitrary JavaScript execution on the database server. The sink is blind — the response is always {"status":"template_saved"} — so detection relies on response delay.

Confidence 64% (Medium)

+50 Confirmed (strong signal)
+6 Inferred from timing / HTTP 500 (no direct echo)
+3 Reproduced by 3 payload(s)
+5 Stable 2xx response

Proof of Concept

Reproduce:

```
curl -i -X POST 'http://localhost/template_engine.php/save' --data
'JSONBODY:{"body":"","filter":{"$where":"var s=Date.now();while(Date.now()-s<5000){};return
true;"}}'
```

Evidence (live response):

```
Proof: response took 4.0s while a non-vulnerable request returns in well under 1s — the server
executed the injected delay (blind time-based injection; no data is echoed). Response body:
{"status":"template_saved"}
```

Remediation

Use parameterized queries / prepared statements — never build queries by string concatenation.

- Replace interpolated SQL with bound parameters (PDO/mysqli prepared statements, '?'/named placeholders).
- For NoSQL, cast user input to the expected scalar type and reject array/operator inputs ((string)$x, type checks).
- Apply least-privilege DB credentials; the web user should not own DDL or admin rights.
- Add allowlist validation for structural elements that cannot be parameterized (column/table names, ORDER BY).

References: CWE-89 · OWASP: SQL Injection Prevention Cheat Sheet · CWE-943 (NoSQL)

---

[2] GGC-VULN-D0C · OUTDATED_COMPONENT — apache 2.4.58 · CVE-2024-38475

LIKELY CVSS 9,8 Critical · CWE-1104 Use of Unmaintained/Vulnerable Third-Party Component · Priority P0 (Fix immediately)

CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

Location: HTTP Server / X-Powered-By banner

Impact: RCE (Direct)

Flow: The server exposes apache 2.4.58, affected by CVE-2024-38475 (fixed in 2.4.60).

Confidence 43% (Low)

+18 Likely (weak signal only)
+20 Direct detection (outdated-component)
+5 Stable 2xx response

Proof of Concept

Reproduce:

```
curl -i 'template_engine.php'
```

Evidence (live response):

```
Server banner 'Apache/2.4.58' matches CVE-2024-38475 (CWE-22, Critical): mod_rewrite improper
output escaping → URL mapped to filesystem (RCE / source disclosure). Affected: version < 2.4.60.
VERSION-INDICATED — confirm the actual build; a backported distro patch may leave the ba…
```

Remediation

Upgrade the flagged component to a fixed release; version-indicated — verify against your actual (possibly backported) build.

- Upgrade to the fixed version listed for the CVE (or a distro build with the backported patch).
- Confirm the real installed version — a backported security fix can leave the banner version unchanged (avoid false positives).
- Suppress the version banner (ServerTokens Prod / expose_php=Off / remove X-Powered-By) to reduce fingerprinting.
- Track dependencies with SCA (composer audit / npm audit / Dependabot) and patch on a schedule.

References: CWE-1104 · OWASP A06:2021 Vulnerable & Outdated Components · NVD

---

[3] GGC-VULN-50D · OUTDATED_COMPONENT — lodash@4.17.15 · CVE-2021-23337

LIKELY CVSS 9,8 Critical · CWE-1104 Use of Unmaintained/Vulnerable Third-Party Component · Priority P0 (Fix immediately)

`CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H`

Location: npm lockfile: package-lock.json

Impact: RCE (Direct)

Flow: lodash@4.17.15 is a known-vulnerable dependency (CVE-2021-23337).

Confidence 49% (Low)

+18 Likely (weak signal only)
+20 Direct detection (outdated-component)
+6 Expected indicator reflected
+5 Stable 2xx response

Proof of Concept

Reproduce:

```
curl -i -X POST 'tests\package-lock.json' --data 'CVE-2021-23337'
```

Evidence (live response):

```
npm package lodash@4.17.15 (in package-lock.json) matches CVE-2021-23337 (CWE-77, High): Command
injection via template(). Fixed in 4.17.21 — upgrade to >= 4.17.21.
```

Remediation

Upgrade the flagged component to a fixed release; version-indicated — verify against your actual (possibly backported) build.

- Upgrade to the fixed version listed for the CVE (or a distro build with the backported patch).
- Confirm the real installed version — a backported security fix can leave the banner version unchanged (avoid false positives).
- Suppress the version banner (ServerTokens Prod / expose_php=Off / remove X-Powered-By) to reduce fingerprinting.
- Track dependencies with SCA (composer audit / npm audit / Dependabot) and patch on a schedule.

References: CWE-1104 · OWASP A06:2021 Vulnerable & Outdated Components · NVD

---

[4] GGC-SSTI-796 · SSTI — body

CONFIRMED CVSS 9,8 Critical · CWE-1336 Server-Side Template Injection · Priority P0 (Fix immediately)

`CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H`

Location: POST /save (JSON body field 'body') → GET ?action=admin_render (rendered output)

Impact: RCE (Chained)

Flow: User-supplied 'body' field from the JSON POST to /save is written to /tmp/ggsec_templates/user_1.txt without any sanitization. When GET ?action=admin_render is requested, the file content is read back and passed through preg_replace_callback matching /\{\{(.*?)\}\}/ which feeds the captured expression into eval('return ' . $m[1] . ';'). Any PHP expression inside {{…}} is executed server-side. This is a second-order SSTI: injection is stored at one endpoint and triggered at another.

Confidence 98% (High)

+50 Confirmed (strong signal)
+20 Direct detection (ssti-rce-eval)
+15 Damning token in response ("uid=")
+6 Expected indicator reflected
+2 Reproduced by 2 payload(s)
+5 Stable 2xx response

Proof of Concept

Reproduce:

```
curl -i -X POST 'template_engine.php?action=admin_render' --data
'RENDER:seed={{system('id')}}|store=/save|storefield=body|render=/?action=admin_render|indicator=uid=|login=/?a
```

Evidence (live response):

```
uid=33(www-data) gid=33(www-data) groups=33(www-data) {"rendered_output":"uid=33(www-data)
gid=33(www-data) groups=33(www-data)"}
```

Remediation

Never build templates from user input — pass user data as bound variables to a sandboxed engine.

- Keep templates static; render user data through the engine's context/variables, never concatenate it into the template source.
- Enable the engine's sandbox/auto-escape (e.g. Jinja2 SandboxedEnvironment, Twig sandbox) and disable dangerous tags/functions.
- Validate/allowlist any value that must influence template selection.

References: CWE-1336 · OWASP: Server-Side Template Injection

---

[5] GGC-VULN-637 · OUTDATED_COMPONENT — apache 2.4.58 · CVE-2024-38476

LIKELY CVSS 7,5 High · CWE-1104 Use of Unmaintained/Vulnerable Third-Party Component · Priority P1 (Within 7 days)

`CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H`

Location: HTTP Server / X-Powered-By banner

Impact: DOS (Direct)

Flow: The server exposes apache 2.4.58, affected by CVE-2024-38476 (fixed in 2.4.60).

Confidence 43% (Low)

+18 Likely (weak signal only)
+20 Direct detection (outdated-component)
+5 Stable 2xx response

Proof of Concept

Reproduce:

```
curl -i 'template_engine.php'
```

Evidence (live response):

```
Server banner 'Apache/2.4.58' matches CVE-2024-38476 (CWE-476, High): mod_proxy null-pointer
dereference → DoS via malicious backend response headers. Affected: version < 2.4.60. VERSION-
INDICATED — confirm the actual build; a backported distro patch may leave the banner version…
```

Remediation

Upgrade the flagged component to a fixed release; version-indicated — verify against your actual (possibly backported) build.

- Upgrade to the fixed version listed for the CVE (or a distro build with the backported patch).
- Confirm the real installed version — a backported security fix can leave the banner version unchanged (avoid false positives).
- Suppress the version banner (ServerTokens Prod / expose_php=Off / remove X-Powered-By) to reduce fingerprinting.
- Track dependencies with SCA (composer audit / npm audit / Dependabot) and patch on a schedule.

References: CWE-1104 · OWASP A06:2021 Vulnerable & Outdated Components · NVD

[6] GGC-VULN-91A · OUTDATED_COMPONENT — guzzlehttp/guzzle@6.5.0 · CVE-2022-31090

LIKELY CVSS 7,5 High · CWE-1104 Use of Unmaintained/Vulnerable Third-Party Component · Priority P1 (Within 7 days)

`CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N`

Location: composer lockfile: composer.lock

Impact: INFORMATION_DISCLOSURE (Direct)

Flow: guzzlehttp/guzzle@6.5.0 is a known-vulnerable dependency (CVE-2022-31090).

Confidence 49% (Low)

+18 Likely (weak signal only)
+20 Direct detection (outdated-component)
+6 Expected indicator reflected
+5 Stable 2xx response

Proof of Concept

Reproduce:

```
curl -i -X POST 'tests\composer.lock' --data 'CVE-2022-31090'
```

Evidence (live response):

```
composer package guzzlehttp/guzzle@6.5.0 (in composer.lock) matches CVE-2022-31090 (CWE-200, Medium): Cross-domain cookie leakage on redirect. Fixed in 6.5.8 — upgrade to >= 6.5.8.
```

Remediation

Upgrade the flagged component to a fixed release; version-indicated — verify against your actual (possibly backported) build.

- Upgrade to the fixed version listed for the CVE (or a distro build with the backported patch).
- Confirm the real installed version — a backported security fix can leave the banner version unchanged (avoid false positives).
- Suppress the version banner (ServerTokens Prod / expose_php=Off / remove X-Powered-By) to reduce fingerprinting.
- Track dependencies with SCA (composer audit / npm audit / Dependabot) and patch on a schedule.

References: CWE-1104 · OWASP A06:2021 Vulnerable & Outdated Components · NVD

[7] GGC-VULN-AD0 · OUTDATED_COMPONENT — symfony/http-kernel@v5.4.10 · CVE-2022-24894

LIKELY CVSS 7,5 High · CWE-1104 Use of Unmaintained/Vulnerable Third-Party Component · Priority P1 (Within 7 days)

`CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N`

Location: composer lockfile: composer.lock

Impact: INFORMATION_DISCLOSURE (Direct)

Flow: symfony/http-kernel@v5.4.10 is a known-vulnerable dependency (CVE-2022-24894).

Confidence 49% (Low)

+18 Likely (weak signal only)
+20 Direct detection (outdated-component)
+6 Expected indicator reflected
+5 Stable 2xx response

Proof of Concept

Reproduce:

```
curl -i -X POST 'tests\composer.lock' --data 'CVE-2022-24894'
```

Evidence (live response):

```
composer package symfony/http-kernel@v5.4.10 (in composer.lock) matches CVE-2022-24894 (CWE-200, Medium): Sensitive header/cache-key information disclosure. Fixed in 5.4.20 — upgrade to >= 5.4.20.
```

Remediation

Upgrade the flagged component to a fixed release; version-indicated — verify against your actual (possibly backported) build.

- Upgrade to the fixed version listed for the CVE (or a distro build with the backported patch).
- Confirm the real installed version — a backported security fix can leave the banner version unchanged (avoid false positives).
- Suppress the version banner (ServerTokens Prod / expose_php=Off / remove X-Powered-By) to reduce fingerprinting.
- Track dependencies with SCA (composer audit / npm audit / Dependabot) and patch on a schedule.

References: CWE-1104 · OWASP A06:2021 Vulnerable & Outdated Components · NVD

---

[8] GGC-VULN-606 · OUTDATED_COMPONENT — lodash@4.17.15 · CVE-2020-8203

LIKELY CVSS 7,5 High · CWE-1104 Use of Unmaintained/Vulnerable Third-Party Component · Priority P1 (Within 7 days)

`CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N`

Location: npm lockfile: package-lock.json

Impact: INFORMATION_DISCLOSURE (Direct)

Flow: lodash@4.17.15 is a known-vulnerable dependency (CVE-2020-8203).

Confidence 49% (Low)

+18 Likely (weak signal only)
+20 Direct detection (outdated-component)
+6 Expected indicator reflected
+5 Stable 2xx response

Proof of Concept

Reproduce:

```
curl -i -X POST 'tests\package-lock.json' --data 'CVE-2020-8203'
```

Evidence (live response):

```
npm package lodash@4.17.15 (in package-lock.json) matches CVE-2020-8203 (CWE-1321, High):
Prototype pollution via zipObjectDeep/set. Fixed in 4.17.19 — upgrade to >= 4.17.19.
```

Remediation

Upgrade the flagged component to a fixed release; version-indicated — verify against your actual (possibly backported) build.

- Upgrade to the fixed version listed for the CVE (or a distro build with the backported patch).
- Confirm the real installed version — a backported security fix can leave the banner version unchanged (avoid false positives).
- Suppress the version banner (ServerTokens Prod / expose_php=Off / remove X-Powered-By) to reduce fingerprinting.
- Track dependencies with SCA (composer audit / npm audit / Dependabot) and patch on a schedule.

References: CWE-1104 · OWASP A06:2021 Vulnerable & Outdated Components · NVD

---

[9] GGC-VULN-463 · OUTDATED_COMPONENT — jquery@3.4.1 · CVE-2020-11022

LIKELY CVSS 7,5 High · CWE-1104 Use of Unmaintained/Vulnerable Third-Party Component · Priority P1 (Within 7 days)

`CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N`

Location: npm lockfile: package-lock.json

Impact: INFORMATION_DISCLOSURE (Direct)

Flow: jquery@3.4.1 is a known-vulnerable dependency (CVE-2020-11022).

Confidence 49% (Low)

+18 Likely (weak signal only)
+20 Direct detection (outdated-component)
+6 Expected indicator reflected
+5 Stable 2xx response

Proof of Concept

Reproduce:

```
curl -i -X POST 'tests\package-lock.json' --data 'CVE-2020-11022'
```

Evidence (live response):

```
npm package jquery@3.4.1 (in package-lock.json) matches CVE-2020-11022 (CWE-79, Medium): XSS via passing HTML from untrusted sources to DOM manipulation methods. Fixed in 3.5.0 — upgrade to >= 3.5.0.
```

Remediation

Upgrade the flagged component to a fixed release; version-indicated — verify against your actual (possibly backported) build.

- Upgrade to the fixed version listed for the CVE (or a distro build with the backported patch).
- Confirm the real installed version — a backported security fix can leave the banner version unchanged (avoid false positives).
- Suppress the version banner (ServerTokens Prod / expose_php=Off / remove X-Powered-By) to reduce fingerprinting.
- Track dependencies with SCA (composer audit / npm audit / Dependabot) and patch on a schedule.

References: CWE-1104 · OWASP A06:2021 Vulnerable & Outdated Components · NVD