GGSec Cortex · GG Advanced IT Security

Security Assessment Report

GGSec Cortex v0.9 Beta · AI-Guided DAST · Context-Aware SAST · Target: smuggle_probe.php · Generated: 2026-07-10 21:12

CONFIDENTIAL

Executive Summary

GGSec Cortex identified 2 confirmed and 4 likely vulnerabilities. Every confirmed finding is evidence-backed (live proof-of-concept or a baseline-difference control), so false positives are minimised. **Remediation is recommended on the normal maintenance cycle.**

Regex-SAST (LLM-free): scanned 1 file(s), 0 candidate sink(s) by pattern. Heuristic pass — findings are Potential (verify); intended as a CI filter / LLM false-negative safety net.

Security Rating	**C · HIGH (2 High)**
Max CVSS (v3.1)	**8,1 (High)**
Severity distribution	Critical: 0 High: 2 Medium: 4 Low: 0
Compromise probability	Moderate (~40-60%)
Confirmed findings	**2**
Likely (needs review)	**4**
Business impact	**Account / administrator takeover**
Exploitation complexity	Low — reachable by a remote attacker
Likelihood	High — confirmed with a live proof-of-concept
Scan	SAST 0s · DAST 88s · 6 requests

Assessment Scope

Target	smuggle_probe.php
Base URL	http://172.26.203.184:8888
Endpoints tested	1
Total requests	6
HTTP methods	SMUGGLE
Vulnerability classes	1
Authentication	Unauthenticated
Scan duration	SAST 0s · DAST 88s
Completed	2026-07-10 21:12

Scan Timeline

21:12:51 ● **Scan started**
21:12:51 ○ SAST replay (cached plan)
21:12:51 ○ DAST probing started
21:12:51 ○ GGC-VULN-CF6 confirmed — REQUEST_SMUGGLING CL.TE
21:12:51 ○ GGC-VULN-24F confirmed — REQUEST_SMUGGLING TE.CL
21:12:51 ○ DAST completed (88s, 6 requests)
21:12:52 ● **Report generated**

Risk Matrix

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

Endpoint Summary

Endpoint	Requests	Findings	Risk
http://172.26.203.184:8888/	6	REQUEST_SMUGGLING	High

Remediation Plan

ID	Finding	CVSS	Priority	SLA
GGC-VULN-CF6	REQUEST_SMUGGLING — CL.TE	8,9	P1	Within 7 days
GGC-VULN-24F	REQUEST_SMUGGLING — TE.CL	8,9	P1	Within 7 days

Findings (6)

[1] GGC-VULN-CF6 · REQUEST_SMUGGLING — CL.TE

CONFIRMED CVSS 8,9 High · CWE-444 HTTP Request Smuggling (front-end/back-end desync) · Priority P1 (Within 7 days)

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

The front-end and back-end disagree on where a request ends (ambiguous Content-Length/Transfer-Encoding), letting an attacker smuggle a second request past the front-end — request hijacking, control bypass, or cache poisoning.

Compliance: A03:2021 – Injection · PCI-DSS v4.0 Req 6.2.4 · ISO/IEC 27001:2022 A.8.28 · GDPR Art. 32 (security of processing)
MITRE ATT&CK: T1190 Exploit Public-Facing Application

Location: smuggle: 172.26.203.184

Impact: AUTH_BYPASS (Direct)

Access: Unauthenticated

Evidence: @ offset 56 — via smuggle-timing-differential: [SMUGGLE] Confirmed HTTP desync (timing-differential) — CL.TE against 172.26.203.184:8888 (native confidence 92%, detecti…

Flow: The front-end and back-end disagree on request boundaries (CL.TE); a crafted Content-Length/Transfer-Encoding mismatch smuggles a second request past the front-end — request hijacking, security-control bypass, cache poisoning, or credential capture.

Evidence strength: High (92/100)

+92 Native desync-engine confidence (timing-differential)

Proof of Concept

Reproduce:

```
curl -i -X POST 'http://172.26.203.184:8888/' --data 'CL.TE'
```

Evidence (live response):

```
[SMUGGLE] Confirmed HTTP desync (timing-differential) — CL.TE against 172.26.203.184:8888 (native confidence 92%, detection timing-differential, timing 8007ms, delta 8006ms). Timing-differential desync (CL.TE): baseline=1ms vs desync-probe=8007ms (~timeout 8000ms). Smuggle marker…
```

Remediation

Reject ambiguous HTTP messages and make the front-end and back-end parse requests identically.

- Reject any request that carries BOTH Content-Length and Transfer-Encoding, or duplicate/conflicting CL or TE headers (respond 400, don't forward).

- Normalize/dechunk requests at the front-end and re-emit a single unambiguous framing to the back-end; disable unsupported Transfer-Encoding variants (identity, obfuscated, folded).
- Ensure the reverse proxy/CDN and origin server use the same HTTP parser and version (prefer HTTP/2 end-to-end to eliminate downgrade desync).
- Disable unsafe back-end connection reuse: close the upstream connection after any malformed or ambiguously-framed request instead of keeping it in the pool.
- Patch and reconfigure the reverse proxy / origin (known CL.TE/TE.CL/TE.TE/0.CL issues are fixed in current releases); add WAF rules that drop conflicting framing headers.

References: CWE-444 · OWASP: HTTP Request Smuggling · PortSwigger: HTTP request smuggling / desync attacks

---

[2] GGC-VULN-24F · REQUEST_SMUGGLING — TE.CL

CONFIRMED CVSS 8,9 High · CWE-444 HTTP Request Smuggling (front-end/back-end desync) · Priority P1 (Within 7 days)

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

The front-end and back-end disagree on where a request ends (ambiguous Content-Length/Transfer-Encoding), letting an attacker smuggle a second request past the front-end — request hijacking, control bypass, or cache poisoning.

Compliance: A03:2021 – Injection · PCI-DSS v4.0 Req 6.2.4 · ISO/IEC 27001:2022 A.8.28 · GDPR Art. 32 (security of processing)
MITRE ATT&CK: T1190 Exploit Public-Facing Application

Location: smuggle: 172.26.203.184

Impact: AUTH_BYPASS (Direct)

Access: Unauthenticated

Evidence: @ offset 56 — via smuggle-timing-differential: [SMUGGLE] Confirmed HTTP desync (timing-differential) — TE.CL against 172.26.203.184:8888 (native confidence 80%, detecti…

Flow: The front-end and back-end disagree on request boundaries (TE.CL); a crafted Content-Length/Transfer-Encoding mismatch smuggles a second request past the front-end — request hijacking, security-control bypass, cache poisoning, or credential capture.

Evidence strength: Medium (80/100)

+80 Native desync-engine confidence (timing-differential)

Proof of Concept

Reproduce:

```
curl -i -X POST 'http://172.26.203.184:8888/' --data 'TE.CL'
```

Evidence (live response):

```
[SMUGGLE] Confirmed HTTP desync (timing-differential) — TE.CL against 172.26.203.184:8888 (native confidence 80%, detection timing-differential, timing 5024ms, delta 5023ms). Time-based confirmation (TE.CL): smuggled GET /?sleep=5 delayed the round-trip to 5024ms (baseline=1ms) â…
```

Remediation

Reject ambiguous HTTP messages and make the front-end and back-end parse requests identically.

- Reject any request that carries BOTH Content-Length and Transfer-Encoding, or duplicate/conflicting CL or TE headers (respond 400, don't forward).
- Normalize/dechunk requests at the front-end and re-emit a single unambiguous framing to the back-end; disable unsupported Transfer-Encoding variants (identity, obfuscated, folded).
- Ensure the reverse proxy/CDN and origin server use the same HTTP parser and version (prefer HTTP/2 end-to-end to eliminate downgrade desync).
- Disable unsafe back-end connection reuse: close the upstream connection after any malformed or ambiguously-framed request instead of keeping it in the pool.
- Patch and reconfigure the reverse proxy / origin (known CL.TE/TE.CL/TE.TE/0.CL issues are fixed in current releases); add WAF rules that drop conflicting framing headers.

References: CWE-444 · OWASP: HTTP Request Smuggling · PortSwigger: HTTP request smuggling / desync attacks

---

[3] GGC-VULN-A18 · REQUEST_SMUGGLING — 0.CL (3 accepted variants)

POTENTIAL CVSS 4,6 Medium (Base 5,4 Medium) · CWE-444 HTTP Request Smuggling — marker-leak / protocol ambiguity (unverified desync) · Priority P2 (Within 30 days)

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

The front-end and back-end disagree on where a request ends (ambiguous Content-Length/Transfer-Encoding), letting an attacker smuggle a second request past the front-end — request hijacking, control bypass, or cache poisoning.

Compliance: A03:2021 – Injection · PCI-DSS v4.0 Req 6.2.4 · ISO/IEC 27001:2022 A.8.28 · GDPR Art. 32 (security of processing)
MITRE ATT&CK: T1190 Exploit Public-Facing Application

Location: smuggle: 172.26.203.184

Impact: INFORMATION_DISCLOSURE (Conditional)

Access: Unauthenticated

Evidence: @ offset 71 — via smuggle-marker-leak: …arker-leak / parser-differential (likely desync — verify) — 0.CL (3 accepted variants) against 172.26.203.184:8888 (native confidence 75%, detecti…

Flow: The 0.CL (3 accepted variants) framing was accepted and a smuggle marker leaked into a subsequent response (protocol ambiguity / likely desync). A real parsing discrepancy, but the full request-hijack impact is unverified — confirm manually before treating as exploitable.

Evidence strength: Medium (75/100)

+75 Native desync-engine confidence (marker-leak)

Proof of Concept

Reproduce:

```
curl -i -X POST 'http://172.26.203.184:8888/' --data '0.CL (3 accepted variants)'
```

Evidence (live response):

```
[SMUGGLE] Marker-leak / parser-differential (likely desync — verify) — 0.CL (3 accepted variants) against 172.26.203.184:8888 (native confidence 75%, detection marker-leak, timing 1ms, delta 1ms). 0.CL smuggle marker leaked into a subsequent response via gadget '/' â€” Content-Le…
```

Remediation

Reject ambiguous HTTP messages and make the front-end and back-end parse requests identically.

- Reject any request that carries BOTH Content-Length and Transfer-Encoding, or duplicate/conflicting CL or TE headers (respond 400, don't forward).
- Normalize/dechunk requests at the front-end and re-emit a single unambiguous framing to the back-end; disable unsupported Transfer-Encoding variants (identity, obfuscated, folded).
- Ensure the reverse proxy/CDN and origin server use the same HTTP parser and version (prefer HTTP/2 end-to-end to eliminate downgrade desync).
- Disable unsafe back-end connection reuse: close the upstream connection after any malformed or ambiguously-framed request instead of keeping it in the pool.
- Patch and reconfigure the reverse proxy / origin (known CL.TE/TE.CL/TE.TE/0.CL issues are fixed in current releases); add WAF rules that drop conflicting framing headers.

References: CWE-444 · OWASP: HTTP Request Smuggling · PortSwigger: HTTP request smuggling / desync attacks

---

[4] GGC-VULN-79D · REQUEST_SMUGGLING — CL.CL

POTENTIAL CVSS 4,6 Medium (Base 5,4 Medium) · CWE-444 HTTP Request Smuggling — marker-leak / protocol ambiguity (unverified desync) · Priority P2 (Within 30 days)

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

The front-end and back-end disagree on where a request ends (ambiguous Content-Length/Transfer-Encoding), letting an attacker smuggle a second request past the front-end — request hijacking, control bypass, or cache poisoning.

Compliance: A03:2021 – Injection · PCI-DSS v4.0 Req 6.2.4 · ISO/IEC 27001:2022 A.8.28 · GDPR Art. 32 (security of processing)
MITRE ATT&CK: T1190 Exploit Public-Facing Application

Location: smuggle: 172.26.203.184

Impact: INFORMATION_DISCLOSURE (Conditional)

Access: Unauthenticated

Evidence: @ offset 71 — via smuggle-marker-leak: …arker-leak / parser-differential (likely desync — verify) — CL.CL against 172.26.203.184:8888 (native confidence 75%, detecti…

Flow: The CL.CL framing was accepted and a smuggle marker leaked into a subsequent response (protocol ambiguity / likely desync). A real parsing discrepancy, but the full request-hijack impact is unverified — confirm manually before treating as exploitable.

Evidence strength: Medium (75/100)

+75 Native desync-engine confidence (marker-leak)

Proof of Concept

Reproduce:

```
curl -i -X POST 'http://172.26.203.184:8888/' --data 'CL.CL'
```

Evidence (live response):

```
[SMUGGLE] Marker-leak / parser-differential (likely desync — verify) — CL.CL against
172.26.203.184:8888 (native confidence 75%, detection marker-leak, timing 0ms, delta -1ms).
Smuggled marker leaked into subsequent response (CL.CL) without clear timing delta (baseline=1ms,
probe…
```

Remediation

Reject ambiguous HTTP messages and make the front-end and back-end parse requests identically.

- Reject any request that carries BOTH Content-Length and Transfer-Encoding, or duplicate/conflicting CL or TE headers (respond 400, don't forward).
- Normalize/dechunk requests at the front-end and re-emit a single unambiguous framing to the back-end; disable unsupported Transfer-Encoding variants (identity, obfuscated, folded).
- Ensure the reverse proxy/CDN and origin server use the same HTTP parser and version (prefer HTTP/2 end-to-end to eliminate downgrade desync).
- Disable unsafe back-end connection reuse: close the upstream connection after any malformed or ambiguously-framed request instead of keeping it in the pool.
- Patch and reconfigure the reverse proxy / origin (known CL.TE/TE.CL/TE.TE/0.CL issues are fixed in current releases); add WAF rules that drop conflicting framing headers.

References: CWE-444 · OWASP: HTTP Request Smuggling · PortSwigger: HTTP request smuggling / desync attacks

---

[5] GGC-VULN-DA4 · REQUEST_SMUGGLING — CL.TE (3 accepted variants)

POTENTIAL CVSS 4,6 Medium (Base 5,4 Medium) · CWE-444 HTTP Request Smuggling — marker-leak / protocol ambiguity (unverified desync) · Priority P2 (Within 30 days)

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

The front-end and back-end disagree on where a request ends (ambiguous Content-Length/Transfer-Encoding), letting an attacker smuggle a second request past the front-end — request hijacking, control bypass, or cache poisoning.

Compliance: A03:2021 – Injection · PCI-DSS v4.0 Req 6.2.4 · ISO/IEC 27001:2022 A.8.28 · GDPR Art. 32 (security of processing)
MITRE ATT&CK: T1190 Exploit Public-Facing Application

Location: smuggle: 172.26.203.184

Impact: INFORMATION_DISCLOSURE (Conditional)

Access: Unauthenticated

Evidence: @ offset 71 — via smuggle-marker-leak: …arker-leak / parser-differential (likely desync — verify) — CL.TE (3 accepted variants) against 172.26.203.184:8888 (native confidence 75%, detecti…

Flow: The CL.TE (3 accepted variants) framing was accepted and a smuggle marker leaked into a subsequent response (protocol ambiguity / likely desync). A real parsing discrepancy, but the full request-hijack impact is unverified — confirm manually before treating as exploitable.

Evidence strength: Medium (75/100)

+75 Native desync-engine confidence (marker-leak)

Proof of Concept

Reproduce:

```
curl -i -X POST 'http://172.26.203.184:8888/' --data 'CL.TE (3 accepted variants)'
```

Evidence (live response):

```
[SMUGGLE] Marker-leak / parser-differential (likely desync — verify) — CL.TE (3 accepted variants)
against 172.26.203.184:8888 (native confidence 75%, detection marker-leak, timing 0ms, delta
-1ms). Smuggled marker leaked into subsequent response (CL.TE (tab-after-colon)) without…
```

Remediation

Reject ambiguous HTTP messages and make the front-end and back-end parse requests identically.

- Reject any request that carries BOTH Content-Length and Transfer-Encoding, or duplicate/conflicting CL or TE headers (respond 400, don't forward).
- Normalize/dechunk requests at the front-end and re-emit a single unambiguous framing to the back-end; disable unsupported Transfer-Encoding variants (identity, obfuscated, folded).
- Ensure the reverse proxy/CDN and origin server use the same HTTP parser and version (prefer HTTP/2 end-to-end to eliminate downgrade desync).
- Disable unsafe back-end connection reuse: close the upstream connection after any malformed or ambiguously-framed request instead of keeping it in the pool.
- Patch and reconfigure the reverse proxy / origin (known CL.TE/TE.CL/TE.TE/0.CL issues are fixed in current releases); add WAF rules that drop conflicting framing headers.

References: CWE-444 · OWASP: HTTP Request Smuggling · PortSwigger: HTTP request smuggling / desync attacks

[6] GGC-VULN-F0B · REQUEST_SMUGGLING — TE.TE (3 accepted variants)

POTENTIAL CVSS 4,6 Medium (Base 5,4 Medium) · CWE-444 HTTP Request Smuggling — marker-leak / protocol ambiguity (unverified desync) · Priority P2 (Within 30 days)

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

The front-end and back-end disagree on where a request ends (ambiguous Content-Length/Transfer-Encoding), letting an attacker smuggle a second request past the front-end — request hijacking, control bypass, or cache poisoning.

Compliance: A03:2021 – Injection · PCI-DSS v4.0 Req 6.2.4 · ISO/IEC 27001:2022 A.8.28 · GDPR Art. 32 (security of processing)
MITRE ATT&CK: T1190 Exploit Public-Facing Application

Location: smuggle: 172.26.203.184

Impact: INFORMATION_DISCLOSURE (Conditional)

Access: Unauthenticated

Evidence: @ offset 71 — via smuggle-marker-leak: …arker-leak / parser-differential (likely desync — verify) — TE.TE (3 accepted variants) against 172.26.203.184:8888 (native confidence 75%, detecti…

Flow: The TE.TE (3 accepted variants) framing was accepted and a smuggle marker leaked into a subsequent response (protocol ambiguity / likely desync). A real parsing discrepancy, but the full request-hijack impact is unverified — confirm manually before treating as exploitable.

Evidence strength: Medium (75/100)

+75 Native desync-engine confidence (marker-leak)

Proof of Concept

Reproduce:

```
curl -i -X POST 'http://172.26.203.184:8888/' --data 'TE.TE (3 accepted variants)'
```

Evidence (live response):

```
[SMUGGLE] Marker-leak / parser-differential (likely desync — verify) — TE.TE (3 accepted variants) against 172.26.203.184:8888 (native confidence 75%, detection marker-leak, timing 0ms, delta -1ms). Smuggled marker leaked into subsequent response (TE.TE) without clear timing delt…
```

Remediation

Reject ambiguous HTTP messages and make the front-end and back-end parse requests identically.

- Reject any request that carries BOTH Content-Length and Transfer-Encoding, or duplicate/conflicting CL or TE headers (respond 400, don't forward).
- Normalize/dechunk requests at the front-end and re-emit a single unambiguous framing to the back-end; disable unsupported Transfer-Encoding variants (identity, obfuscated, folded).
- Ensure the reverse proxy/CDN and origin server use the same HTTP parser and version (prefer HTTP/2 end-to-end to eliminate downgrade desync).
- Disable unsafe back-end connection reuse: close the upstream connection after any malformed or ambiguously-framed request instead of keeping it in the pool.
- Patch and reconfigure the reverse proxy / origin (known CL.TE/TE.CL/TE.TE/0.CL issues are fixed in current releases); add WAF rules that drop conflicting framing headers.

References: CWE-444 · OWASP: HTTP Request Smuggling · PortSwigger: HTTP request smuggling / desync attacks