TiDB Security Assessment

PingCAP

May 1, 2026

Version 1.1

Prepared for:
PingCAP

Prepared by:
Tom Kramkowski
Rob Russell

1 Executive Summary

Synopsis

From June 25th, 2025 to September 4th, 2025, PingCAP engaged NCC Group to conduct a security assessment of TiDB. TiDB is a cloud-native distributed SQL database. Source code was provided for the assessment, and the entire system was in scope. Testing was performed by one (1) consultant across forty-five (45) days of effort against version 8.5.2 of the product.

Scope

NCC Group's evaluation included:

- **Source Code Review**: A broad review of the source code using static analysis tools followed by a targeted manual review of key areas relating to security.
- **Dynamic Testing**: A review of the security posture of an example deployment of TiDB.

Code review was performed locally, and dynamic testing was performed on a dedicated test environment.

Key Findings

The assessment uncovered a set of common application flaws. The most notable findings were:

- The default security policy for communication between TiDB components was weak, which could be abused by a suitably positioned attacker to intercept inter-component traffic and attack the individual components.
- Static analysis of the source code repositories indicated that the project's CI pipelines likely did not run common useful static analysis tools such as `gosec`, `govulncheck`, or `cargo-audit`.

Strategic Recommendations

- Review the CI configuration for TiDB and integrate common static analysis tools in order to spot potential issues earlier and simplify future code audits.
- Consider requiring users to opt-in to insecure configurations, rather than having insecure options configured by default.

Retest Update (4/27/2026)

From April 21st through April 27th, 2026, NCC Group performed a retest of six (6) of the eight (8) vulnerabilities identified during the initial assessment. Testing was performed by one (1) consultant, across five (5) days of effort, against version 8.5.6 of the product. Four (4) of the six (6) in-scope findings were still present by default in out-of-the-box deployments. However, relevant security controls are available for customer configuration, and NCC Group verified the effectiveness of these controls when properly configured during the retest. Because of this, and because PingCAP documentation provides best practice recommendations to customers, these four findings have been marked **Risk Accepted**.

Of the remaining two (2) in-scope findings, one (1) was found to be **Fixed** and one (1) **Partially Fixed**.

For details on the current status of each vulnerability, see the Table of Findings, and for detailed information about the retest results, see the "Retest Results" section of each relevant finding in the Finding Details section of this report.

2 Dashboard

Target Data		Engagement Data	
Name	TiDB	**Type**	Source Code Review and Dynamic Testing
Type	Distributed Database Software	**Method**	White-box
Platforms	Rust and Go	**Dates**	2025-06-25 to 2025-09-04
Environment	Local and Testing	**Consultants**	1
		Level of Effort	45 person-days

Targets

github.com/pingcap/tidb	#: f43a13324440f92209e2a9f04c0bbe9cf763978d (v8.5.2)
github.com/pingcap/tidb-ctl	#: 9c165585d38e16591d459f21c0c82b9518747aba
github.com/pingcap/tidb-tools	#: a611536696221844eeb8ec5a9a2e90ca87223a54 (v8.5.2)
github.com/pingcap/tidb-binlog	#: 552cffbb46230d7b4a41995379423ae9d2351a47 (v8.5.0-alpha)
github.com/pingcap/tidb-operator	#: 3aaf3e378abcdba2b40111adc0910dcf7260ac88
github.com/pingcap/tidb-dashboard	#: 259f8f007e16396a43e0292ff6a8184173fb8abb (v8.5.2)
github.com/pingcap/tiflash	#: 189d3bf630c41c302fdcc44ec055c376a5753b0a (v8.5.2)
github.com/pingcap/tiflow	#: 536a87813274d3169488e2b324ef0d033d854bd8 (v8.5.2)
github.com/pingcap/tiflow-operator	#: 9ab11c525e3693cc92199e344f0f394233800821
github.com/pingcap/monitoring	#: ff668f5e9e6650c634b6202a31aef25e17d74e8f (v8.5.2)
github.com/pingcap/ng-monitoring	#: b3f04b6b1d51afbaccdab2a27533e35da882c95f (v8.5.2)
github.com/tikv/tikv	#: a150e4569fda1c64763fda297f4e09775759de4a (v8.5.2)
github.com/tikv/pd	#: 4cd009c4db3c15215341a96521dd53e53c55e5bd (v8.5.2)
github.com/tikv/migration	#: e056406992cc8bde52171493320e201dc7231e66
github.com/pingcap/tiup	#: 7b34205dc6f9297894aba43442dd64ea4ec79420
github.com/pingcap/diag	#: d2d5237c2dd36d44627bb9f5a8a9d75b67b37e30 (v1.6.0)
github.com/PingCAP-QE/artifacts	#: 6021e90e29eb48909914e9fc25b716ba19d9df6e
github.com/tikv/client-go	#: 9b1e2a6652f9b6093a70118054ea754f6302856f
github.com/tikv/rust-rocksdb	#: 755d152ec4eb443039296ceecf9a072a1142e014
github.com/tikv/raft-engine	#: 392f5e66f8286dc1b6d7cf69f2bc20ed72d40123
TiDB Deployment	10.50.0.162 10.50.0.165 10.50.0.171 10.50.0.174 10.50.0.178 10.50.0.214 10.50.0.223 10.50.0.244 10.50.0.245

Finding Breakdown

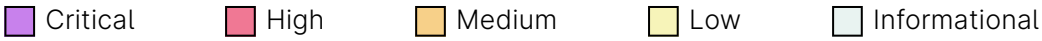
	Original Assessment	Remaining
Critical issues	0	0
High issues	0	0
Medium issues	0	0
Low issues	8	7
Informational issues	0	0

Category Breakdown

Access Controls	3	
Cryptography	1	
Data Exposure	1	
Patching	2	

Component Breakdown

Prometheus	1	
Source Code	2	
TiDB	2	
Various	2	



3 Table of Findings

For each finding, NCC Group uses a composite risk score that takes into account the severity of the risk, application's exposure and user population, technical difficulty of exploitation, and other factors.

Prometheus

Title	Status	ID	Risk
Prometheus Exposed	Reported	QPR	Low

Source Code

Title	Status	ID	Risk
Known Vulnerabilities in Pinned Rust Dependencies	Reported	P4K	Low
Known Vulnerabilities in Pinned Go Dependencies	Partially Fixed	276	Low
Many Unaddressed Gosec Linter Warnings	Fixed	FX3	Low

TiDB

Title	Status	ID	Risk
'pprof' Endpoints Exposed	Risk Accepted	BP2	Low
Internal API Information Disclosure	Risk Accepted	P39	Low

Various

Title	Status	ID	Risk
Status/Administrative Endpoints Require No Authentication By Default	Risk Accepted	77Q	Low
Encryption Not Enforced By Default	Risk Accepted	VJM	Low

4 Finding Details – Prometheus

Low

Prometheus Exposed

Overall Risk	Low	**Finding ID**	NCC-E022582-QPR
Impact	Low	**Component**	Prometheus
Exploitability	High	**Category**	Access Controls
		Status	Reported

Impact

An attacker with access to the internal network would be able to access the Prometheus web interface and affect the state of the Prometheus services.

Description

The Prometheus service running on the monitoring host of the example TiDB deployment was exposed on port 9090, did not require authentication to access and was not configured to require TLS.

This appeared to be a conscious design decision as various components of TiDB appeared to communicate with the service.

However, exposure of this service increased the potential attack surface in case of a compromise of the network forming a TiDB deployment.

```
State  Recv-Q Send-Q Local Address:Port  Peer
↳ Address:PortProcess
LISTEN 0      128          0.0.0.0:22         0.0.0.0:*    users:
↳ (("sshd",pid=1103,fd=3))
LISTEN 0      32768    10.50.0.245:9094       0.0.0.0:*    users:
↳ (("alertmanager",pid=722,fd=3))
LISTEN 0      32768              *:9115             *:*    users:
↳ (("blackbox_export",pid=723,fd=3))
LISTEN 0      32768              *:9093             *:*    users:
↳ (("alertmanager",pid=722,fd=8))
LISTEN 0      32768              *:9090             *:*    users:
↳ (("prometheus",pid=737,fd=7))
LISTEN 0      32768              *:9100             *:*    users:
↳ (("node_exporter",pid=733,fd=3))
LISTEN 0      128             [::]:22            [::]:*    users:
↳ (("sshd",pid=1103,fd=4))
LISTEN 0      32768              *:12020            *:*    users:(("ng-monitoring-
↳ s",pid=754,fd=18))
LISTEN 0      32768              *:3000             *:*    users:(("grafana-
↳ server",pid=725,fd=12))
LISTEN 0      32768              *:2380             *:*    users:(("pd-
↳ server",pid=736,fd=7))
LISTEN 0      32768              *:2379             *:*    users:(("pd-
↳ server",pid=736,fd=8))
```

Figure 1: Output from `netstat -tlnp` on 10.50.0.245

Recommendation

Review whether the Prometheus service needs to be exposed to the network. Where exposure is required, implement appropriate security controls such as authentication and encryption, or consider adopting a zero-trust architecture that leverages an overlay or service mesh network to secure communication between supporting components.

Location

- 10.50.0.245:9090

Retest Results

2026-04-27 – Not Tested

This finding was not within scope for retesting.

5 Finding Details – Source Code

Low

Known Vulnerabilities in Pinned Rust Dependencies

Overall Risk	Low	**Finding ID**	NCC-E022582-P4K
Impact	Low	**Component**	Source Code
Exploitability	Medium	**Category**	Patching
		Status	Reported

Impact

Future code changes or unexplored attack paths may allow an attacker to abuse known vulnerabilities in Rust crates to compromise parts of a TiDB deployment.

Description

Analysis using `cargo-audit` identified several binary crates with pinned dependencies with known vulnerabilities. A manual review of the current codebase did not reveal any direct exploitation paths for these specific issues. However, the presence of these known vulnerabilities may put the software at risk either due to attack paths which may have been missed or due to future code changes which may expose the vulnerabilities.

Additionally, some dependencies were no longer maintained.

The following is a summary of the vulnerable or unmaintained dependencies:

- tikv/tikv
  - openssl 0.10.57 RUSTSEC-2024-0357 RUSTSEC-2025-0022 RUSTSEC-2025-0004
  - remove_dir_all 0.5.3 RUSTSEC-2023-0018
  - time 0.1.43 RUSTSEC-2020-0071
  - ansi_term 0.11.0 RUSTSEC-2021-0139 unmaintained
  - atty 0.2.13 RUSTSEC-2024-0375 unmaintained
  - bcc 0.0.30 RUSTSEC-2024-0383 unmaintained
  - cpuid-bool 0.1.2 RUSTSEC-2021-0064 unmaintained
  - derivative 2.2.0 RUSTSEC-2024-0388 unmaintained
  - instant 0.1.12 RUSTSEC-2024-0384 unmaintained
  - memmap 0.7.0 RUSTSEC-2020-0077 unmaintained
  - mmap 0.1.1 RUSTSEC-2024-0394 unmaintained
  - net2 0.2.37 RUSTSEC-2020-0016 unmaintained
  - paste 1.0.4 RUSTSEC-2024-0436 unmaintained
  - proc-macro-error 1.0.4 RUSTSEC-2024-0370 unmaintained
  - safemem 0.3.2 RUSTSEC-2023-0081 unmaintained
  - serde_cbor 0.11.1 RUSTSEC-2021-0127 unmaintained
  - tempdir 0.3.7 RUSTSEC-2018-0017 unmaintained
  - twoway 0.2.0 RUSTSEC-2021-0146 unmaintained
  - atty 0.2.13 RUSTSEC-2021-0145 unsound
  - mimalloc 0.1.25 RUSTSEC-2022-0094 unsound

    - openssl 0.10.57 RUSTSEC-2023-0072 unsound
- pingcap/tiflash (libs/libprocess_metrics)
    - libc 0.2.154 yanked

Recommendation

Review the affected code to determine whether the vulnerable pinned dependencies can be updated. After making changes, test the software to confirm the updates have not introduced new issues. For dependencies that are no longer maintained, develop a plan to replace them with actively maintained alternatives. Consider incorporating tools such as cargo-audit into the CI pipeline to track newly disclosed vulnerabilities and ensure they are acted on promptly.

Location

- tikv/tikv
- pingcap/tiflash (libs/libprocess_metrics)

Retest Results

2026-04-27 – Not Tested

This finding was not within scope for retesting.

Low

Known Vulnerabilities in Pinned Go Dependencies

Overall Risk	Low	**Finding ID**	NCC-E022582-276
Impact	Low	**Component**	Source Code
Exploitability	Low	**Category**	Patching
		Status	Partially Fixed

Impact

Future code changes or unexplored attack paths may allow an attacker to abuse known vulnerabilities in Go packages to compromise parts of a TiDB deployment.

Description

Analysis using `govulncheck` identified several binary crates with pinned dependencies with known vulnerabilities. A manual review of the codebase did not reveal any direct exploitation paths for these specific issues. However, the presence of these known vulnerabilities may put the software at risk either due to attack paths which may have been missed or due to future code changes which may expose the vulnerabilities.

The following is a summary of the identified vulnerable dependencies:

- pingcap/tidb-binlog
  - github.com/Azure/azure-sdk-for-go/sdk/azidentity v0.12.0 GO-2024-2918
  - github.com/golang/glog v1.0.0 GO-2025-3372
  - github.com/pingcap/tidb v1.1.0-beta.0.20230303060110-ad01574a71b3 GO-2024-3284
  - golang.org/x/net v0.17.0 GO-2024-2687
  - google.golang.org/grpc v1.52.3 GO-2023-2153
  - google.golang.org/protobuf v1.28.1 GO-2024-2611
- pingcap/tiflow-operator
  - github.com/prometheus/client_golang v1.11.0 GO-2022-0322
  - golang.org/x/net v0.0.0-20220722155237-a158d28d115b GO-2023-1571 GO-2023-1571 GO-2024-2687 GO-2024-2687
  - k8s.io/kubernetes v1.24.0 GO-2022-0983 GO-2023-1628 GO-2023-1629 GO-2023-1864 GO-2023-1891 GO-2023-1892 GO-2023-2341 GO-2024-2994 GO-2024-3277 GO-2025-3465 GO-2025-3521 GO-2025-3522 GO-2025-3547
- pingcap/tidb-ctl
  - github.com/pingcap/tidb v1.1.0-beta.0.20200519125814-6098373c11a9 GO-2024-3284
- pingcap/diag
  - golang.org/x/crypto v0.14.0 GO-2023-2402
  - golang.org/x/crypto v0.14.0 GO-2025-3487
  - golang.org/x/net v0.17.0 GO-2024-2687
- pingcap/ng-monitoring
  - github.com/pingcap/tidb v1.1.0-beta.0.20240409103151-ec8e27a7ed23 GO-2024-3284

- pingcap/tidb-operator
  - github.com/mholt/archiver/v3 v3.5.1 GO-2024-2698
- pingcap/tiflow
  - github.com/pingcap/tidb v1.1.0-beta.0.20231117065153-a4f85c356873 GO-2024-3284
  - github.com/pingcap/tidb v1.1.0-beta.0.20240228123331-27ce02afd2e3 GO-2024-3284
  - golang.org/x/net v0.18.0 GO-2024-2687
  - golang.org/x/net v0.21.0 GO-2024-2687
- pingcap/tidb-dashboard
  - golang.org/x/net v0.23.0 GO-2025-3595
  - github.com/golang-jwt/jwt v3.2.1+incompatible GO-2025-3553
- pingcap/tidb-tools
  - github.com/golang/glog v1.2.0 GO-2025-3372
  - github.com/pingcap/tidb v1.1.0-beta.0.20241119124618-50b5cd27d413 GO-2024-3284
  - google.golang.org/grpc v1.64.0 GO-2024-2978
- tikv/migration
  - github.com/Azure/azure-sdk-for-go/sdk/azidentity v0.12.0 GO-2024-2918
  - github.com/pingcap/tidb v1.1.0-beta.0.20220222031143-5988d0b2f46e GO-2024-3284
  - github.com/pingcap/tidb v1.1.0-beta.0.20231124053542-069631e2ecfe GO-2024-3284
  - go.etcd.io/etcd v0.5.0-alpha.5.0.20191023171146-3cf2f69b5738 GO-2024-2527 GO-2024-2528 GO-2024-2529 GO-2024-2530
  - go.opentelemetry.io/contrib/instrumentation/google.golang.org/grpc/otelgrpc v0.25.0 GO-2023-2331
  - golang.org/x/net v0.0.0-20210405180319-a5a99cb37ef4 GO-2023-1571 GO-2024-2687
  - golang.org/x/net v0.17.0 GO-2024-2687
  - golang.org/x/net v0.33.0 GO-2025-3503 GO-2025-3595
  - google.golang.org/protobuf v1.30.0 GO-2024-2611
  - google.golang.org/protobuf v1.31.0 GO-2024-2611
- tikv/pd
  - golang.org/x/net v0.25.0 GO-2025-3595
  - github.com/golang-jwt/jwt/v4 v4.4.2 GO-2025-3553 GO-2024-3250

Recommendation

Review the affected code to determine whether the pinned dependencies with known vulnerabilities can be updated. After making changes, test the software to confirm the updates have not introduced new issues. For dependencies that are no longer maintained, develop a plan to replace them with actively maintained alternatives. Consider utilizing tools such as govulncheck as part of CI to keep track of new vulnerabilities so that they can be acted on promptly.

Location

- pingcap/diag
- pingcap/ng-monitoring
- pingcap/tidb-binlog
- pingcap/tidb-ctl
- pingcap/tidb-dashboard
- pingcap/tidb-operator
- pingcap/tidb-tools
- pingcap/tiflow
- pingcap/tiflow-operator
- tikv/migration
- tikv/pd

Retest Results

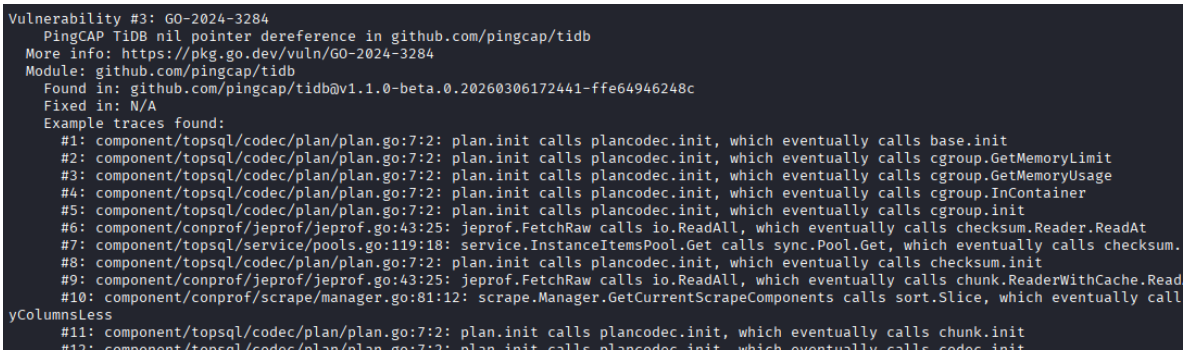
2026-04-23 – Partially Fixed

The following (updated) repositories listed as “primary components” were provided for verification.

- https://github.com/pingcap/ng-monitoring/releases/tag/v8.5.6
- https://github.com/pingcap/tidb-dashboard/releases/tag/v8.5.6
- https://github.com/pingcap/tidb-tools/releases/tag/v8.5.5
- https://github.com/pingcap/tiflow/releases/tag/v8.5.6
- https://github.com/tikv/pd/releases/tag/v8.5.6

While many of the originally reported instances appeared to be fixed, some were still present, as shown below:

`https://github.com/pingcap/ng-monitoring/releases/tag/v8.5.6`



`https://github.com/pingcap/tidb-tools/releases/tag/v8.5.5`

```
Vulnerability #5: GO-2024-2978
    Private tokens could appear in logs if context containing gRPC metadata is
    logged in google.golang.org/grpc
  More info: https://pkg.go.dev/vuln/GO-2024-2978
  Module: google.golang.org/grpc
    Found in: google.golang.org/grpc@v1.64.0
    Fixed in: google.golang.org/grpc@v1.64.1
    Example traces found:
      #1: pkg/check/table_structure.go:472:15: check.briefColumnInfo.String calls fmt.Fprintln, which e

Your code is affected by 5 vulnerabilities from 4 modules.
This scan also found 7 vulnerabilities in packages you import and 7
vulnerabilities in modules you require, but your code doesn't appear to call
these vulnerabilities.
Use '-show verbose' for more details.
```

`https://github.com/pingcap/tiflow/releases/tag/v8.5.6`

```
Vulnerability #3: GO-2024-3284
    PingCAP TiDB nil pointer dereference in github.com/pingcap/tidb
  More info: https://pkg.go.dev/vuln/GO-2024-3284
  Module: github.com/pingcap/tidb
    Found in: github.com/pingcap/tidb@v1.1.0-beta.0.20260312160659-adb287a22464
    Fixed in: N/A
    Example traces found:
      #1: cdc/processor/processor.go:1068:6: processor.ddlHandler.Run calls errgroup.Group.Go, which eventually calls admin.CheckIndicesCount
      #2: cdc/processor/processor.go:1068:6: processor.ddlHandler.Run calls errgroup.Group.Go, which eventually calls admin.CheckRecordAndIndex
      #3: dm/pkg/conn/mockdb.go:152:38: conn.NewCluster calls session.BootstrapSession, which eventually calls admin.CheckRecordAndIndex
      #4: cdc/processor/processor.go:1068:6: processor.ddlHandler.Run calls errgroup.Group.Go, which eventually calls admin.CheckRecordAndIndex
      #5: engine/jobmaster/dm/checkpoint/agent.go:25:2: checkpoint.init calls executor.init, which calls admin.init
      #6: dm/pkg/conn/mockdb.go:152:38: conn.NewCluster calls session.BootstrapSession, which eventually calls affinity.CreateGroupsIfNotExists
      #7: dm/pkg/conn/mockdb.go:152:38: conn.NewCluster calls session.BootstrapSession, which eventually calls affinity.DeleteGroupsWithRetry
      #8: cdc/processor/processor.go:1068:6: processor.ddlHandler.Run calls errgroup.Group.Go, which eventually calls affinity.GetAllGroupStates
      #9: cdc/entry/schema_test_helper.go:70:26: entry.NewSchemaTestHelperWithReplicaConfig calls testkit.NewTestKit, which eventually calls affinity.InitMana
      #10: engine/jobmaster/dm/checkpoint/agent.go:25:2: checkpoint.init calls executor.init, which calls affinity.init
      #11: dm/pkg/conn/mockdb.go:152:38: conn.NewCluster calls session.BootstrapSession, which eventually calls aggfuncs.Build
      #12: dm/pkg/conn/mockdb.go:152:38: conn.NewCluster calls session.BootstrapSession, which eventually calls aggfuncs.BuildWindowFunctions
      #13: cdc/processor/processor.go:1068:6: processor.ddlHandler.Run calls errgroup.Group.Go, which eventually calls aggfuncs.NewAggPartialResultMapper
      #14: dm/pkg/conn/mockdb.go:152:38: conn.NewCluster calls session.BootstrapSession, which eventually calls aggfuncs.NewSerializeHelper
      #15: cdc/processor/processor.go:1068:6: processor.ddlHandler.Run calls errgroup.Group.Go, which eventually calls aggfuncs.approxCountDistinctFinal.Appen
Result2Chunk
      #16: cdc/processor/processor.go:1068:6: processor.ddlHandler.Run calls errgroup.Group.Go, which eventually calls aggfuncs.approxCountDistinctOriginal.Up
```

This finding is therefore considered **Partially Fixed**.

Low

Many Unaddressed Gosec Linter Warnings

Overall Risk	Low	**Finding ID**	NCC-E022582-FX3
Impact	Low	**Component**	Source Code
Exploitability	Undetermined	**Category**	Other
		Status	Fixed

Impact

Developers and security reviewers may overlook real security issues as a result of noise from Gosec.

Description

A number of the Go projects forming the TiDB solution produced an overwhelming amount of Gosec linter warnings, indicating that Gosec was likely not part of the CI pipeline and highlighting areas of potential improvement for the codebase. The warnings were reviewed to determine their security impact, and no such impact was identified. However, the volume of warnings caused some review overhead, taking valuable review time away from other areas of the codebase.

In many cases, the warnings related to potential bugs or code which could have been rewritten more simply and correctly. In other cases, the warnings were false positives.

In the example below, `fmt.Sprintf` is used to build a SQL query using a table name taken from a constant. This triggers Gosec, as code like this often represents a SQL injection opportunity. In this case, the warning was a false positive. The normal recommendation for code such as this is to use a query builder, although in this circumstance, it can be argued that formatting a SQL query using constants is a relatively safe operation compared to the operations where query builders are normally beneficial.

```
100 func (bo *GenericOptions) SetTikvGCLifeTime(ctx context.Context, db *sql.DB, gcTime
    ↳string) error {
101   sql := fmt.Sprintf("update %s set variable_value = ? where variable_name = ?",
      ↳constants.TidbMetaTable)
102   _, err := db.ExecContext(ctx, sql, gcTime, constants.TikvGCVariable)
103   if err != nil {
104     return fmt.Errorf("set cluster %s %s failed, sql: %s, err: %v", bo,
        ↳constants.TikvGCVariable, sql, err)
105   }
106   return nil
107 }
```

Figure 2: pingcap/tidb-operator: `cmd/backup-manager/app/util/generic.go`

Recommendation

Run the Gosec linter on the affected repositories and review the output. Modify code to avoid triggering linter warnings where possible without affecting the meaning of the code. In the remaining instances, annotate the code with `// nolint: gosec` and include an explanatory comment justifying silencing the Gosec linter.

Integrate Gosec into the CI pipeline to catch code changes which would introduce new warnings.

In the case of pingcap/tidb-operator (`cmd/backup-manager/app/util/generic.go`) consider if a SQL query builder package would be more appropriate to form the SQL queries.

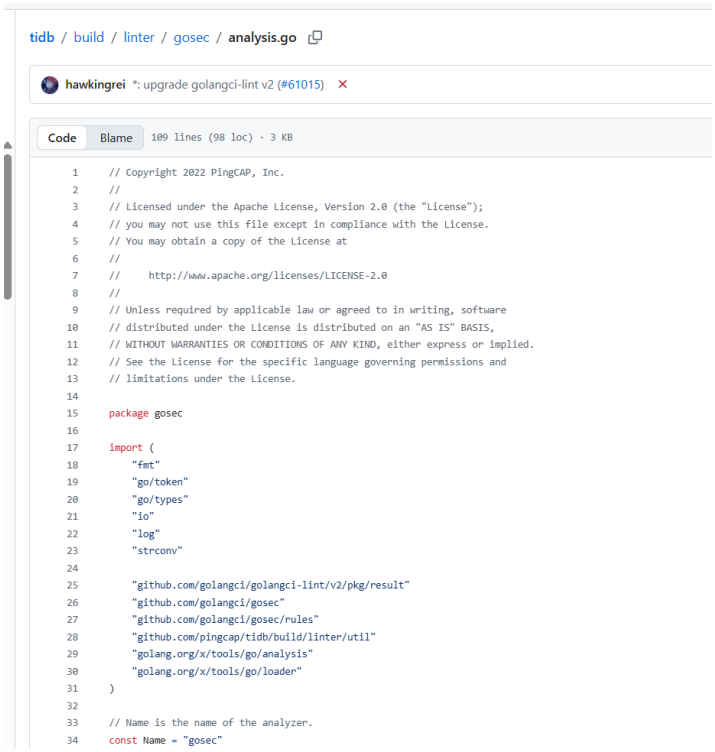
Location

- pingcap/diag
- pingcap/monitoring
- pingcap/ng-monitoring
- pingcap/tidb
- pingcap/tidb-binlog
- pingcap/tidb-ctl
- pingcap/tidb-dashboard
- pingcap/tidb-operator
- pingcap/tidb-tools
- pingcap/tiflow
- pingcap/tiflow-operator
- pingcap/tiup
- tikv/client-go
- tikv/migration
- tikv/pd

Retest Results

2026-04-23 – Fixed

The Gosec linter is now integrated into the CI pipeline:



This finding is therefore considered **Fixed**.

6 Finding Details – TiDB

Low

‘pprof’ Endpoints Exposed

Overall Risk	Low	**Finding ID**	NCC-E022582-BP2
Impact	Low	**Component**	TiDB
Exploitability	High	**Category**	Access Controls
		Status	Risk Accepted

Impact

An attacker with access to the internal TiDB network may be able to use information gathered from `pprof` to perform further targeted attacks.

Description

Some services forming the TiDB deployment exposed the `pprof` web interface. This performance profiling package exposes detailed information about the running Go program as a web interface, which can be useful for debugging purposes. However, this information may be useful to an attacker during subsequent exploitation attempts. This interface did not require authentication and was not configured with TLS.

```
[ec2-user@ip-10-50-0-245 ~]$ curl -s http://localhost:12020/debug/pprof/cmdline | tr '\0' ' ';
↳ echo
bin/ng-monitoring-server --config /tidb-deploy/prometheus-9090/conf/ngmonitoring.toml
```

Figure 3: Example pprof endpoint

As TiDB is an open source project, the impact of leaking certain details such as function names is limited. However, `pprof` can be used to dump goroutine stack traces and object memory addresses which can, for example, help an attacker verify the success or progress of remote code execution attacks.

Recommendation

Disable the `pprof` web interfaces except for debug builds of the applications.

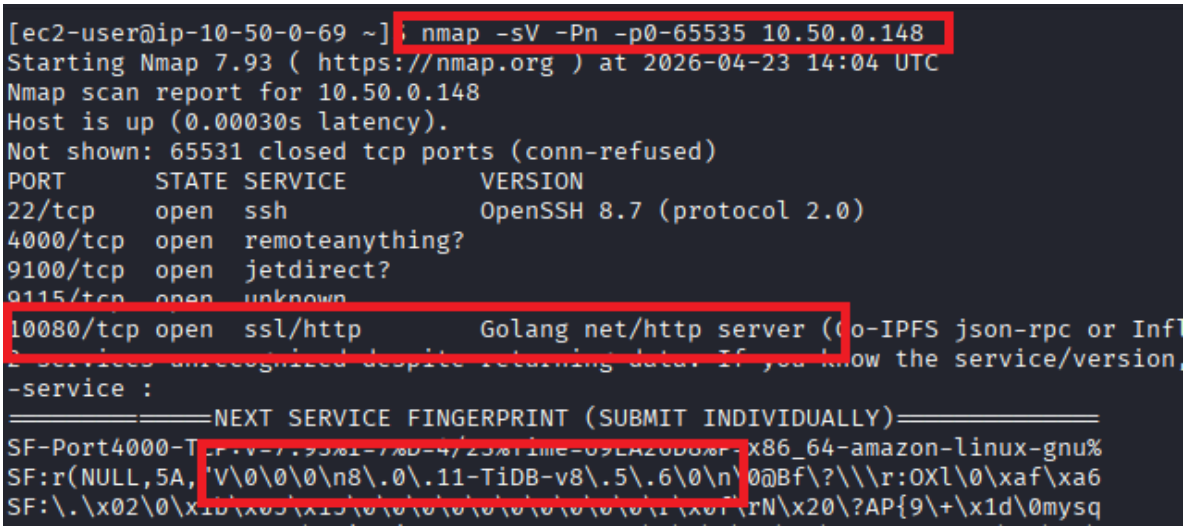
Location

- http://10.50.0.178:10080/debug/pprof/
- http://10.50.0.171:10080/debug/pprof/

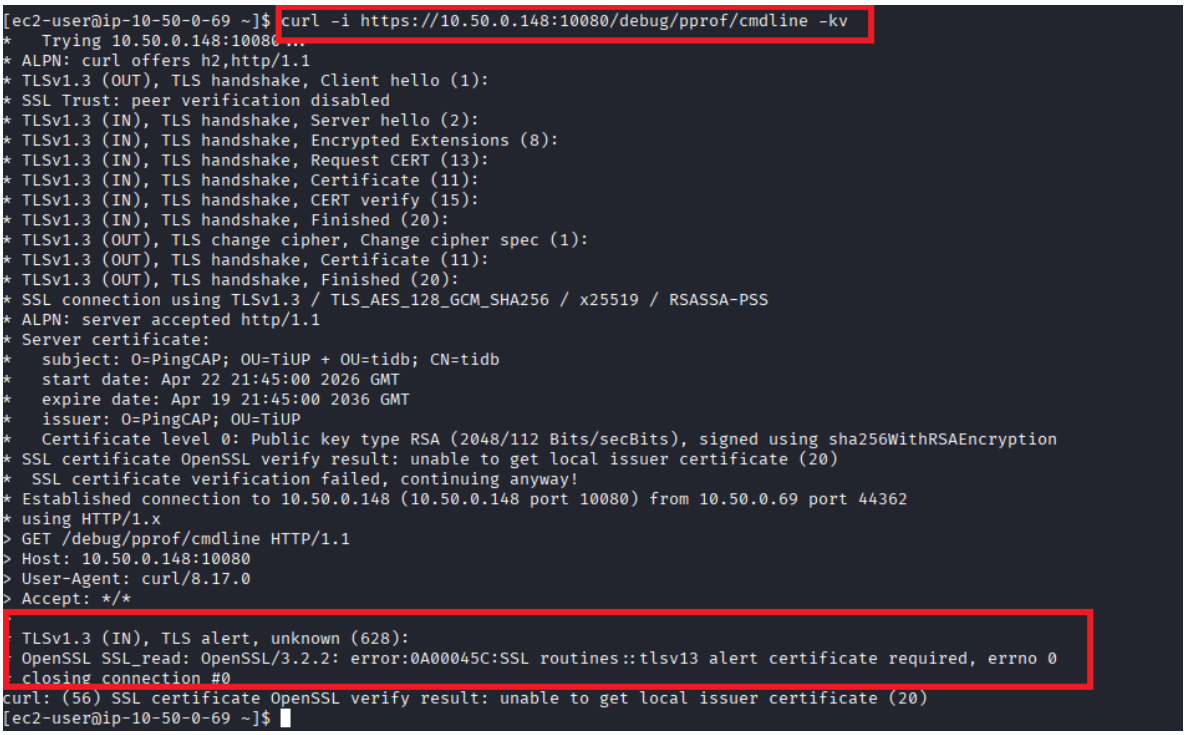
Retest Results

2026-04-23 – Not Fixed

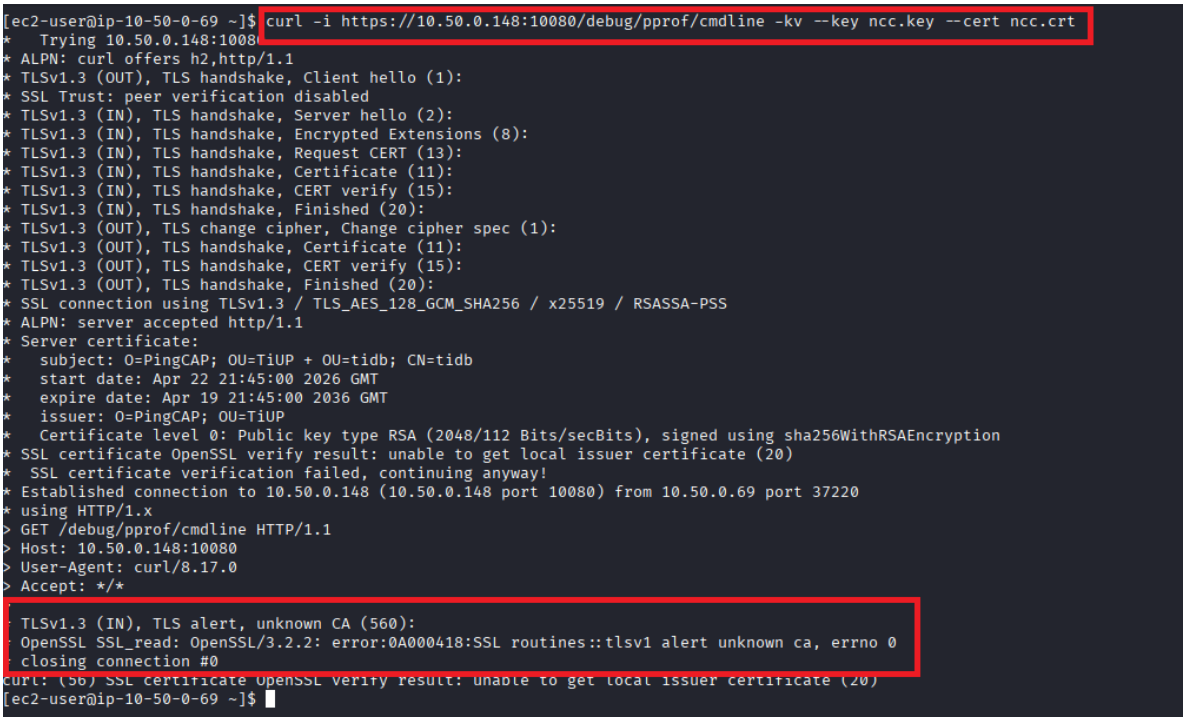
NCC Group scanned the provided TiDB instance and noted that the affected port was open, but this time was protected by SSL/TLS:



An attempt was made to access the previously exposed endpoint, which resulted in the following error indicating that an mTLS client certificate was expected:



NCC Group attempted to connect again after generating and providing a self-signed client certificate, resulting in a new error indicating that the provided client certificate was not trusted:



This indicated that the endpoint was now enforcing mTLS. However, based on discussions with Pingcap, the provided test environment represents a deployment of the product where the relevant security controls have been correctly configured, and these issues are still present by default in out-of-the-box deployments.

As these security controls are expected to be configured by the customer, best practice recommendations are provided in Pingcap documentation, and because the results of this re-test have confirmed that the controls are effective when properly configured, NCC Group has marked this finding as **Risk Accepted**.

Low

Internal API Information Disclosure

Overall Risk	Low	**Finding ID**	NCC-E022582-P39
Impact	Low	**Component**	TiDB
Exploitability	Medium	**Category**	Data Exposure
		Status	Risk Accepted

Impact

An attacker with access to the internal network hosting a TiDB deployment may access internal information, such as the configuration of TiDB.

Description

Backend APIs exposed by TiDB produced certain detailed internal information, such as the configuration. This information was exposed with no authentication by default. These insecure defaults put deployments at risk unless a conscious effort was made by administrators to implement mutual TLS (see finding "Encryption Not Enforced By Default").

An example of the kind of information disclosed is listed below. It includes details of other hosts forming the cluster, as well as local file paths.

```
{
  "host": "0.0.0.0",
  "advertise-address": "10.50.0.171",
  "port": 4000,
  "cors": "",
  "store": "tikv",
  "path": "10.50.0.245:2379,10.50.0.174:2379,10.50.0.244:2379",
  "socket": "/tmp/tidb-4000.sock",
  "lease": "45s",
  "split-table": true,
  "token-limit": 1000,
  "temp-dir": "/tmp/tidb",
  "tmp-storage-path": "/tmp/1000_tidb/MC4wLjAuMDo0MDAwLzAuMC4wLjA6MTAwODA=/tmp-storage",
  ...
  "log": {
    ...
    "file": {
      "filename": "/tidb-deploy/tidb-4000/log/tidb.log",
      ...
    },
    "slow-query-file": "/tidb-deploy/tidb-4000/log/tidb_slow_query.log",
    ...
  },
  ...
}
```

Figure 4: Excerpt from a request to `http://10.50.0.171:10080/config`

Recommendation

Require explicit opt-in for insecure configurations. Review the recommendations for finding "Status/Administrative Endpoints Require No Authentication By Default".

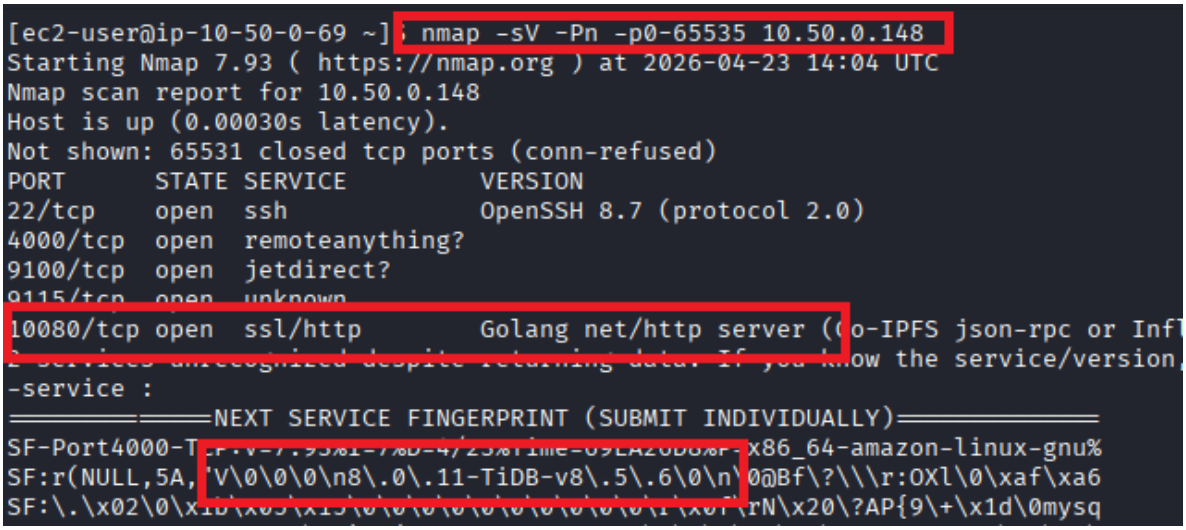
Location

- http://10.50.0.171:10080/config
- http://10.50.0.178:10080/config

Retest Results

2026-04-23 – Not Fixed

NCC Group scanned the provided TiDB instance and noted that the affected port was open, but this time was protected by SSL/TLS:



An attempt was made to access the previously exposed endpoint, which resulted in the following error indicating that an mTLS client certificate was expected:

```
[ec2-user@ip-10-50-0-69 ~]$ curl -i https://10.50.0.148:10080/config -k
curl: (56) SSL certificate OpenSSL verify result: unable to get local issuer certificate (20)
[ec2-user@ip-10-50-0-69 ~]$ curl -i https://10.50.0.148:10080/config -kv
*   Trying 10.50.0.148:10080...
* ALPN: curl offers h2,http/1.1
* TLSv1.3 (OUT), TLS handshake, Client hello (1):
* SSL Trust: peer verification disabled
* TLSv1.3 (IN), TLS handshake, Server hello (2):
* TLSv1.3 (IN), TLS handshake, Encrypted Extensions (8):
* TLSv1.3 (IN), TLS handshake, Request CERT (13):
* TLSv1.3 (IN), TLS handshake, Certificate (11):
* TLSv1.3 (IN), TLS handshake, CERT verify (15):
* TLSv1.3 (IN), TLS handshake, Finished (20):
* TLSv1.3 (OUT), TLS change cipher, Change cipher spec (1):
* TLSv1.3 (OUT), TLS handshake, Certificate (11):
* TLSv1.3 (OUT), TLS handshake, Finished (20):
* SSL connection using TLSv1.3 / TLS_AES_128_GCM_SHA256 / x25519 / RSASSA-PSS
* ALPN: server accepted http/1.1
* Server certificate:
*   subject: O=PingCAP; OU=TiUP + OU=tidb; CN=tidb
*   start date: Apr 22 21:45:00 2026 GMT
*   expire date: Apr 19 21:45:00 2036 GMT
*   issuer: O=PingCAP; OU=TiUP
*   Certificate level 0: Public key type RSA (2048/112 Bits/secBits), signed using sha256WithRSAEncryption
* SSL certificate OpenSSL verify result: unable to get local issuer certificate (20)
*  SSL certificate verification failed, continuing anyway!
* Established connection to 10.50.0.148 (10.50.0.148 port 10080) from 10.50.0.69 port 47890
* using HTTP/1.x
> GET /config HTTP/1.1
> Host: 10.50.0.148:10080
> User-Agent: curl/8.17.0
> Accept: */*
>
* TLSv1.3 (IN), TLS alert, unknown (628):
* OpenSSL SSL_read: OpenSSL/3.2.2: error:0A00045C:SSL routines::tlsv13 alert certificate required, errno 0
* closing connection #0
curl: (56) SSL certificate OpenSSL verify result: unable to get local issuer certificate (20)
[ec2-user@ip-10-50-0-69 ~]$
```

NCC Group attempted to connect again after generating and providing a self-signed client certificate, resulting in a new error indicating that the provided client certificate was not trusted:

```
[ec2-user@ip-10-50-0-69 ~]$ curl https://10.50.0.148:10080/config -vk --cert ncc.crt --key ncc.key
*   Trying 10.50.0.148:10080...
* ALPN: curl offers h2,http/1.1
* TLSv1.3 (OUT), TLS handshake, Client hello (1):
* SSL Trust: peer verification disabled
* TLSv1.3 (IN), TLS handshake, Server hello (2):
* TLSv1.3 (IN), TLS handshake, Encrypted Extensions (8):
* TLSv1.3 (IN), TLS handshake, Request CERT (13):
* TLSv1.3 (IN), TLS handshake, Certificate (11):
* TLSv1.3 (IN), TLS handshake, CERT verify (15):
* TLSv1.3 (IN), TLS handshake, Finished (20):
* TLSv1.3 (OUT), TLS change cipher, Change cipher spec (1):
* TLSv1.3 (OUT), TLS handshake, Certificate (11):
* TLSv1.3 (OUT), TLS handshake, CERT verify (15):
* TLSv1.3 (OUT), TLS handshake, Finished (20):
* SSL connection using TLSv1.3 / TLS_AES_128_GCM_SHA256 / x25519 / RSASSA-PSS
* ALPN: server accepted http/1.1
* Server certificate:
*   subject: O=PingCAP; OU=TiUP + OU=tidb; CN=tidb
*   start date: Apr 22 21:45:00 2026 GMT
*   expire date: Apr 19 21:45:00 2036 GMT
*   issuer: O=PingCAP; OU=TiUP
*   Certificate level 0: Public key type RSA (2048/112 Bits/secBits), signed using sha256WithRSAEncryption
* SSL certificate OpenSSL verify result: unable to get local issuer certificate (20)
*  SSL certificate verification failed, continuing anyway!
* Established connection to 10.50.0.148 (10.50.0.148 port 10080) from 10.50.0.69 port 35052
* using HTTP/1.x
> GET /config HTTP/1.1
> Host: 10.50.0.148:10080
> User-Agent: curl/8.17.0
> Accept: */*
>
* TLSv1.3 (IN), TLS alert, unknown CA (560):
* OpenSSL SSL_read: OpenSSL/3.2.2: error:0A000418:SSL routines::tlsv1 alert unknown ca, errno 0
* closing connection #0
curl: (56) SSL certificate OpenSSL verify result: unable to get local issuer certificate (20)
[ec2-user@ip-10-50-0-69 ~]$
```

This indicated that the endpoint was now enforcing mTLS. However, based on discussions with Pingcap, the provided test environment represents a deployment of the product where the relevant security controls have been correctly configured, and these issues are still present by default in out-of-the-box deployments.

As these security controls are expected to be configured by the customer, best practice recommendations are provided in Pingcap documentation, and because the results of this re-test have confirmed that the controls are effective when properly configured, NCC Group has marked this finding as **Risk Accepted**.

7 Finding Details – Various

Low

Status/Administrative Endpoints Require No Authentication By Default

Overall Risk	Low	**Finding ID**	NCC-E022582-77Q
Impact	Medium	**Component**	Various
Exploitability	Medium	**Category**	Access Controls
		Status	Risk Accepted

Impact

An attacker with access to the internal network of a default TiDB deployment can gather information and control portions of the deployment without needing to first gather credentials.

Description

TiDB and its constituent components exposed metrics and control endpoints which did not require authentication. Authentication enforcement was only possible with the deployment of mutual TLS which, along with TLS, were not configured by default (Refer to finding "Encryption Not Enforced By Default").

Recommendation

Consider only allowing access to these endpoints from the local host by default, thereby limiting the attack surface of a stock deployment of TiDB. Alternatively, if the metrics endpoints are required to be accessed by other cluster hosts, consider requiring administrators deploying TiDB to opt into this insecure configuration.

Refer to finding "Encryption Not Enforced By Default" for more relevant recommendations.

Location

- pingcap/tidb `pkg/server/http_status.go`
- tikv/pd `server/api/router.go`
- tikv/tikv `src/server/status_server/mod.rs`

Retest Results

2026-04-23 – Not Fixed

During the re-test, the affected endpoints were found to be enforcing mTLS. However, based on discussions with Pingcap, the provided test environment represents a deployment of the product where the relevant security controls have been correctly configured, and these issues are still present by default in out-of-the-box deployments.

As these security controls are expected to be configured by the customer, best practice recommendations are provided in Pingcap documentation, and because the results of this re-test have confirmed that the controls are effective when properly configured, NCC Group has marked this finding as **Risk Accepted**.

Low

Encryption Not Enforced By Default

Overall Risk	Low	**Finding ID**	NCC-E022582-VJM
Impact	Medium	**Component**	Various
Exploitability	Medium	**Category**	Cryptography
		Status	Risk Accepted

Impact

An attacker with access to a network running a stock deployment of TiDB and its constituent components would be able to intercept traffic between components.

Description

TiDB, PD and TiKV by default do not utilize TLS to communicate between each-other.[1] This puts their communications at risk of interception and even tampering. These components do support TLS, but it appears that their default configuration does not use it. A lack of any guardrails to either warn or prevent such a default configuration means that a default deployment of TiDB would be at risk of compromise if system administrators did not take extra steps to mitigate this risk.

Recommendation

As TLS requires certificates, it is not easy to configure it by default. However, silently defaulting to unencrypted communication should be avoided. Ideally, unencrypted communication should be an opt-in setting, requiring a conscious configuration decision. In case unencrypted communication is not opted into, but TLS certificates are not provided, the components should produce error messages to indicate a misconfiguration.

At the very least, warning messages should loudly indicate that the deployment is running unencrypted (with an option to silence them in cases where this is intentional due to transport security being handled externally).

Location

- pingcap/tidb `pkg/store/driver/tikv_driver.go`
- tikv/pd `server/server.go`
- tikv/tikv `src/server/server.rs`

Retest Results

2026-04-23 – Not Fixed

During the re-test, the affected services were found to be enforcing TLS, with an example shown below:

1. TiDB Docs: Enable TLS Between TiDB Components: https://docs.pingcap.com/tidb/stable/enable-tls-between-components/

```
Your MySQL connection id is 1637875724
Server version: 8.0.11-TiDB-v8.5.6 TiDB Server (Apache License 2.0) Community Edition, MySQL 8.0 compatible

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]> SHOW STATUS LIKE 'Ssl_version';
+---------------+---------+
| Variable_name | Value   |
+---------------+---------+
| Ssl_version   | TLSv1.3 |
+---------------+---------+
1 row in set (0.111 sec)

MySQL [(none)]> SHOW STATUS LIKE 'Ssl_cipher';
+---------------+------------------------+
| Variable_name | Value                  |
+---------------+------------------------+
| Ssl_cipher    | TLS_AES_128_GCM_SHA256 |
+---------------+------------------------+
1 row in set (0.112 sec)
```

However, based on discussions with Pingcap, the provided test environment represents a deployment of the product where the relevant security controls have been correctly configured, and these issues are still present by default in out-of-the-box deployments. As these security controls are expected to be configured by the customer, best practice recommendations are provided in Pingcap documentation, and because the results of this re-test have confirmed that the controls are effective when properly configured, NCC Group has marked this finding as **Risk Accepted**.

8 Finding Field Definitions

The following sections describe the risk rating and category assigned to issues NCC Group identified.

Risk Scale

NCC Group uses a composite risk score that takes into account the severity of the risk, application's exposure and user population, technical difficulty of exploitation, and other factors. The risk rating is NCC Group's recommended prioritization for addressing findings. Every organization has a different risk sensitivity, so to some extent these recommendations are more relative than absolute guidelines.

Overall Risk

Overall risk reflects NCC Group's estimation of the risk that a finding poses to the target system or systems. It takes into account the impact of the finding, the difficulty of exploitation, and any other relevant factors.

Rating	Description
Critical	Implies an immediate, easily accessible threat of total compromise.
High	Implies an immediate threat of system compromise, or an easily accessible threat of large-scale breach.
Medium	A difficult to exploit threat of large-scale breach, or easy compromise of a small portion of the application.
Low	Implies a relatively minor threat to the application.
Informational	No immediate threat to the application. May provide suggestions for application improvement, functional issues with the application, or conditions that could later lead to an exploitable finding.

Impact

Impact reflects the effects that successful exploitation has upon the target system or systems. It takes into account potential losses of confidentiality, integrity and availability, as well as potential reputational losses.

Rating	Description
High	Attackers can read or modify all data in a system, execute arbitrary code on the system, or escalate their privileges to superuser level.
Medium	Attackers can read or modify some unauthorized data on a system, deny access to that system, or gain significant internal technical information.
Low	Attackers can gain small amounts of unauthorized information or slightly degrade system performance. May have a negative public perception of security.

Exploitability

Exploitability reflects the ease with which attackers may exploit a finding. It takes into account the level of access required, availability of exploitation information, requirements relating to social engineering, race conditions, brute forcing, etc, and other impediments to exploitation.

Rating	Description
High	Attackers can unilaterally exploit the finding without special permissions or significant roadblocks.
Medium	Attackers would need to leverage a third party, gain non-public information, exploit a race condition, already have privileged access, or otherwise overcome moderate hurdles in order to exploit the finding.
Low	Exploitation requires implausible social engineering, a difficult race condition, guessing difficult-to-guess data, or is otherwise unlikely.

Category

NCC Group categorizes findings based on the security area to which those findings belong. This can help organizations identify gaps in secure development, deployment, patching, etc.

Category Name	Description
Access Controls	Related to authorization of users, and assessment of rights.
Auditing and Logging	Related to auditing of actions, or logging of problems.
Authentication	Related to the identification of users.
Configuration	Related to security configurations of servers, devices, or software.
Cryptography	Related to mathematical protections for data.
Data Exposure	Related to unintended exposure of sensitive information.
Data Validation	Related to improper reliance on the structure or values of data.
Denial of Service	Related to causing system failure.
Error Reporting	Related to the reporting of error conditions in a secure fashion.
Patching	Related to keeping software up to date.
Session Management	Related to the identification of authenticated users.
Timing	Related to race conditions, locking, or order of operations.

9 Contact Info

The team from NCC Group has the following primary members:

- Tom Kramkowski – Consultant
  tom.kramkowski@nccgroup.com
- Rob Russell – Retest Consultant
  rob.russell@nccgroup.com
- Lois Herr – Project Manager I
  lois.herr@nccgroup.com
- Tim Friendson – Project Manager II
  tim.friendson@nccgroup.com

The team from PingCAP has the following primary members:

- Mark Donsky
  mark.donsky@pingcap.com
- Oliver Tajiki
  oliver.tajiki@pingcap.com
- Stephen Thorn
  stephen.thorn@pingcap.com