CSAEC

CyberScan — Cyber Server Audit Engine Core

CyberScan Technical Guide

Architecture, assessment workflow, security controls, operations, and maintenance

Version 1.0 | August 2026
Microsoft Windows 10 or later | 64-bit

Document information

Field	Value
Document title	CyberScan Technical Guide
Product	CyberScan - Cyber Server Audit Engine Core
Organisation	CSAEC
Version	1.0
Publication date	August 2026
Document code	CS-TG-EN-1.0
Language	English
Intended audience	Software engineers, IT administrators, cybersecurity professionals, deployment teams, technical partners, and maintainers
Classification	Public distribution

Document control

Version	Date	Owner	Change
1.0	August 2026	CSAEC	Initial Windows publication

Disclaimer

CyberScan provides a preliminary technical assessment and decision-support information. It is not an antivirus, EDR platform, managed SOC, penetration-testing replacement, formal certification tool, regulatory compliance guarantee, continuous-monitoring platform, or automatic remediation system. Findings and AI output require review. Significant changes should be validated by qualified personnel.

Table of contents

This contents list follows the document section order. Page numbers are provided in the footer for navigation.

01 Purpose and scope
02 Intended audience
03 Product overview
04 Windows system requirements
05 High-level architecture
06 Technology stack
07 Application components
08 Desktop application architecture
09 Backend service layer
10 User interface layer
11 Python-to-interface communication
12 Security assessment workflow
13 The twelve security controls
14 Network discovery architecture
15 Port and service assessment
16 Known vulnerability assessment
17 Local web service assessment
18 Result data structure
19 Severity classification
20 Scoring methodology
21 Result caching and interface refresh
22 Offline AI architecture
23 AI model initialisation
24 Optional local Ollama integration
25 Request serialisation and thread safety
26 Internationalisation
27 Language persistence
28 Local data processing
29 Privacy and user-controlled sharing
30 Report generation
31 Email client integration
32 Windows packaging and distribution
33 Dependency management
34 Error handling
35 Logging considerations
36 Security design principles
37 Known limitations
38 Troubleshooting
39 Maintenance guidelines
40 Extension guidelines
41 Glossary

1. Purpose and scope

CyberScan - Cyber Server Audit Engine Core is a Windows desktop security assessment application that performs a structured, local-first review of twelve security controls. This guide documents the verified high-level architecture, responsibilities, result contract, assessment workflow, operational constraints, and extension principles.

> Scope boundary
> The guide describes what the application evaluates. Where exact Windows implementation details are not confirmed, mechanisms remain intentionally implementation-neutral.

Assessment objectives

- Provide a repeatable baseline security posture overview.
- Present individual control outcomes, severity, explanations, and general corrective actions.
- Support preliminary reviews and preparation for cyber-risk questionnaires.
- Keep the user in control of report sharing and external communication.

2. Intended audience

This guide is written for engineers, administrators, cybersecurity practitioners, deployment teams, partners, and maintainers who need to understand product behaviour without assuming unsupported internal mechanisms.

3. Product overview

CyberScan assesses the condition visible at scan time, aggregates the twelve control outcomes into a score from 0 to 100, and retains individual findings as the primary decision inputs. The integrated local AI assistant explains findings and offers general guidance; it does not alter Windows settings.

Positioning

CyberScan is	CyberScan is not
An initial security assessment tool	An antivirus or EDR platform
A baseline security review solution	A penetration-testing replacement
A security posture overview and decision-support application	A compliance guarantee or certification
A preliminary technical assessment	A continuous-monitoring or automatic remediation system

4. Windows system requirements

Requirement area	Supported baseline
Operating system	Microsoft Windows 10 or later, 64-bit edition
Environment	Standard Windows desktop environment
Storage	Sufficient local storage for the application, security components, and local AI model
Permissions	Suitable user permissions for system and network security checks
Hardware	Detailed hardware requirements may vary according to the selected application build and local AI configuration.

Configuration dependency
Some checks may return limited or unavailable results when permissions, local policy, security software, or network filtering restricts visibility.

5. High-level architecture

The application uses a Python desktop backend and a native desktop window hosting an inline HTML, CSS, and JavaScript interface through pywebview. The backend exposes controlled methods to the interface. Assessment components produce structured results; the interface resolves semantic result codes into the selected language and renders cards, score, details, and actions.

Layer	Responsibility	Key boundary
Desktop shell	Hosts the user interface and manages application windows	Does not itself determine security posture
Interface layer	Starts checks, caches results, renders status, score, and actions	Consumes structured backend responses
Backend service layer	Coordinates checks, reports, AI requests, and external tool invocation	Exposes a limited interface to the UI
Assessment components	Evaluate system, network, vulnerability, and configuration indicators	Visibility depends on permissions and environment
Local AI layer	Explains findings through a local model or optional local runtime	Informational output only

6. Technology stack

Technology	Role	Status
Python 3	Application backend and orchestration	Confirmed
pywebview	Desktop window and Python-to-interface bridge	Confirmed
HTML, CSS, JavaScript	Inline user interface without a separate frontend bundle	Confirmed
Nmap / python-nmap	Network discovery, port inspection, and vulnerability-oriented checks	Component availability may depend on build
Wapiti	Selected tests against reachable local web services	Component availability may depend on build
llama-cpp-python	Embedded local language-model execution	Confirmed architecture
Qwen2.5 1.5B Instruct GGUF	Local explanatory model	Confirmed model family and format
Ollama	Optional locally available model runtime	Optional
PyInstaller	Windows application packaging approach	Confirmed tool; exact distribution package may vary

7. Application components

Component	Primary responsibility
Scanner API	Provides callable assessment and support functions to the interface.
Control evaluators	Return structured status, severity, score, and optional finding details.
Result cache	Retains raw results in the interface for redraw and language changes.
Score renderer	Aggregates control scores and displays the overall classification.
Security Wiki	Provides interactive explanatory content and AI-assisted guidance.
Offline LLM service	Initialises and serialises local model requests.
Report and contact workflow	Prepares user-controlled report sharing through the default email client.

8. Desktop application architecture

CyberScan follows a compact desktop architecture: a Python process coordinates assessment logic and supplies the inline interface to a pywebview window. A separate Wiki window may be created on demand and reused while open. This design keeps core control flow within one application boundary while preserving a clear interface contract.

Window lifecycle

- The main window hosts scan controls, status cards, the score, and report actions.
- The Wiki window is created when requested and brought forward if already open.
- Language context and explanatory content are propagated to the Wiki.

9. Backend service layer

The backend provides the Scanner API, coordinates synchronous control execution, locates packaged or system-available assessment components, and returns structured dictionaries rather than display text. Long-running checks can block their individual call; the interface communicates progress to keep the workflow understandable.

Service responsibilities

Responsibility	Design note
Control execution	Each control has a defined callable entry point and stable semantic result contract.
Tool selection	Specialised tools are used only where available and appropriate to the control.
AI mediation	All embedded model calls pass through the controlled AI service.
Sharing	The application prepares a user-visible communication; sending remains a deliberate user action.

10. User interface layer

The interface uses vanilla JavaScript and custom CSS. It orchestrates the twelve modules, maintains the most recent structured results, calculates display summaries, resolves translated text, and presents individual recheck actions.

Interface states

State	Meaning	Expected presentation
Not assessed	No current result is available	Neutral card and scan prompt
In progress	The selected check is executing	Busy indicator and progress message
Completed	A structured result is available	Severity, explanation, and action
Limited / error	The check could not produce a complete result	Clear limitation or troubleshooting message

11. Python-to-interface communication

The interface calls exposed backend methods through the pywebview API bridge. Results use semantic codes rather than pre-rendered text. The active language catalog resolves each code into a status, description, technical context, and general action.

Field	Purpose
success	Indicates whether the check completed sufficiently to return a result.
code	Stable semantic identifier resolved by the interface language catalog.
level	Normalised severity: ok, warn, or bad.
score	Numeric contribution used by the overall score.
params	Optional values used to render contextual text.
hosts / ports / vulns	Optional structured observations for relevant network checks.

12. Security assessment workflow

Process flow

Step	Activity	Output
1	User starts or repeats an assessment.	Control list enters an in-progress state.
2	The interface invokes each backend control.	Structured result per control.
3	Raw results are stored in the interface cache.	Reusable current scan state.
4	The interface resolves language and renders cards.	Severity, explanation, action, and details.
5	The score summary is recalculated.	0-100 score and classification.
6	The user reviews, rechecks, asks the AI, or creates a report.	User-controlled next action.

Point-in-time result
Network state, radio exposure, service availability, and update configuration can change immediately after a scan. Results must be interpreted with their assessment time.

13. Detailed description of the twelve security controls

1. Windows Firewall Status

Purpose and evaluation. Confirms whether host firewall protection is active and appropriately configured at a high level.

Security relevance. A disabled or unsuitable firewall can increase exposure to unsolicited connections.

Presentation. CyberScan returns a structured outcome that the interface presents as a severity, concise explanation, score contribution, and recommended next action.

General corrective action. Enable and review Windows firewall protection using approved organisational settings. Validate exceptions before changing them.

Limitations. The result is a point-in-time configuration indicator and does not prove that every rule is appropriate.

2. Disk Encryption Status

Purpose and evaluation. Evaluates whether storage protection indicates that data at rest is encrypted.

Security relevance. Unencrypted storage can expose business data if a device is lost, stolen, or accessed without authorisation.

Presentation. CyberScan returns a structured outcome that the interface presents as a severity, concise explanation, score contribution, and recommended next action.

General corrective action. Use an organisation-approved disk-encryption configuration and preserve recovery information securely.

Limitations. The check does not assess every key-management, recovery, or escrow practice.

3. Operating System Security and Integrity

Purpose and evaluation. Reviews available indicators of operating-system integrity and security configuration.

Security relevance. Weakened integrity controls can make unauthorised changes harder to prevent or detect.

Presentation. CyberScan returns a structured outcome that the interface presents as a severity, concise explanation, score contribution, and recommended next action.

General corrective action. Apply supported security settings and investigate unexpected changes with qualified personnel.

Limitations. Coverage depends on the Windows edition, permissions, and information available to the application.

4. Local Network Port Exposure

Purpose and evaluation. Uses network-oriented inspection, including Nmap where available, to identify reachable local ports and services.

Security relevance. Unnecessary listening services may increase the attack surface.

Presentation. CyberScan returns a structured outcome that the interface presents as a severity, concise explanation, score contribution, and recommended next action.

General corrective action. Confirm business need, restrict access, and remove or reconfigure unneeded services after impact review.

Limitations. Filtered or closed services and local network conditions can affect visibility.

5. Local Network Device Discovery

Purpose and evaluation. Discovers devices visible on the current local network segment using available network information.

Security relevance. Unknown devices can indicate unmanaged assets, guest access, or a need for inventory review.

Presentation. CyberScan returns a structured outcome that the interface presents as a severity, concise explanation, score contribution, and recommended next action.

General corrective action. Compare results with an authorised asset list and investigate unexplained devices.

Limitations. Discovery is not a complete inventory; segmentation, filtering, and sleeping devices can hide assets.

6. Automatic Screen-Lock Configuration

Purpose and evaluation. Evaluates whether an automatic lock and re-authentication policy is configured.

Security relevance. An unattended, unlocked workstation can permit unauthorised access.

Presentation. CyberScan returns a structured outcome that the interface presents as a severity, concise explanation, score contribution, and recommended next action.

General corrective action. Adopt a suitable inactivity timeout and require authentication on return, consistent with business policy.

Limitations. The check does not assess user behaviour or every policy source.

7. Windows Security Update Configuration

Purpose and evaluation. Reviews the configuration indicators available for Windows security updates.

Security relevance. Delayed security updates can leave known weaknesses unaddressed.

Presentation. CyberScan returns a structured outcome that the interface presents as a severity, concise explanation, score contribution, and recommended next action.

General corrective action. Maintain an approved update process, test business-critical changes, and resolve persistent failures.

Limitations. Configuration status does not prove that every update is installed successfully.

8. Antivirus or Endpoint Protection Status

Purpose and evaluation. Evaluates available indicators of antivirus or endpoint protection status.

Security relevance. Missing, disabled, or outdated protection can reduce detection and response capability.

Presentation. CyberScan returns a structured outcome that the interface presents as a severity, concise explanation, score contribution, and recommended next action.

General corrective action. Enable a supported protection solution, verify updates, and review alerts according to policy.

Limitations. CyberScan is not an antivirus or EDR platform and does not replace their monitoring.

9. Backup Configuration

Purpose and evaluation. Reviews available indicators that a backup configuration exists.

Security relevance. Without usable backups, recovery from ransomware, failure, or accidental deletion may be difficult.

Presentation. CyberScan returns a structured outcome that the interface presents as a severity, concise explanation, score contribution, and recommended next action.

General corrective action. Maintain protected backups and test restoration regularly using approved procedures.

Limitations. Configuration alone does not prove backup completeness, integrity, isolation, or restorability.

10. Wireless and Bluetooth Exposure

Purpose and evaluation. Reviews available indicators of wireless interface and Bluetooth exposure.

Security relevance. Unnecessary discoverability or active interfaces can increase nearby attack opportunities.

Presentation. CyberScan returns a structured outcome that the interface presents as a severity, concise explanation, score contribution, and recommended next action.

General corrective action. Disable unused interfaces and limit discoverability according to operational needs.

Limitations. Radio state and discoverability can change after the assessment.

11. Known Vulnerability Assessment

Purpose and evaluation. Uses vulnerability-oriented checks, including Nmap components where available, against detected services.

Security relevance. Potentially vulnerable services may permit compromise or unauthorised access.

Presentation. CyberScan returns a structured outcome that the interface presents as a severity, concise explanation, score contribution, and recommended next action.

General corrective action. Validate findings, prioritise by exposure and impact, then apply vendor-supported mitigation.

Limitations. Results can include false positives or false negatives and are not proof of exploitability.

12. Local Web Service Vulnerability Assessment

Purpose and evaluation. Uses Wapiti where available to assess locally exposed web services for selected vulnerability classes.

Security relevance. Weak local web services may expose data or enable attacks through the workstation.

Presentation. CyberScan returns a structured outcome that the interface presents as a severity, concise explanation, score contribution, and recommended next action.

General corrective action. Validate findings in an authorised context, then update, reconfigure, or remove affected services.

Limitations. Only reachable services are assessed; coverage is not equivalent to a full web application penetration test.

14. Network discovery architecture

Network discovery determines assets visible from the current workstation and local network context. Nmap may support device discovery and related inspection. Results are influenced by network segmentation, filtering, host availability, interface selection, and the privileges granted to CyberScan.

> Authorisation
> Run network and vulnerability-oriented checks only on systems and networks for which the user has permission.

15. Port and service assessment

Port assessment inspects the configured local target and identifies reachable services at scan time. Open ports are observations, not vulnerabilities by themselves. Business purpose, exposure, access restrictions, service version, and patch state determine risk.

Observation	Interpretation
Open	A service accepted or responded to the relevant probe.
Closed	The target was reachable, but the service did not accept the connection.
Filtered / unknown	Filtering or network behaviour prevented a definitive determination.

16. Known vulnerability assessment

The known vulnerability control may use Nmap vulnerability-oriented components against detected services. It provides indications for validation, not guaranteed vulnerability detection. Script coverage, signatures, service identification, filtering, and target behaviour affect results.

> Validation required
> A potential match may be a false positive, and an absent result may be a false negative. Confirm important findings using vendor information and qualified analysis.

17. Local web service assessment

When a locally exposed web service is reachable, Wapiti may assess selected vulnerability classes. The assessment is constrained to detected and reachable content, configured modules, scan time, and permitted scope. It is not a complete application security review or penetration test.

18. Result data structure

Category	Examples	Handling
Core outcome	success, code, level, score	Required for standard rendering and scoring.
Context parameters	Control-specific values	Used to explain the observed condition.
Network observations	hosts, ports, services	Rendered only where relevant.
Vulnerability observations	candidate findings	Presented for validation with limitations.

19. Severity classification

Level	User meaning	Operational response
OK	The available indicators meet the control baseline.	Maintain the control and review contextual details.
Warning	The result is incomplete, uncertain, or merits improvement.	Investigate and schedule a proportionate corrective action.
High attention	A material weakness may be present.	Validate promptly and obtain qualified assistance where needed.

20. Scoring methodology

The overall score summarises the twelve control scores on a 0-100 scale. Individual findings remain more important than the total alone because operational impacts differ. The score reflects conditions at assessment time and does not guarantee security, certification, or compliance.

Score	Classification	Interpretation
85-100	Good	A sound baseline is indicated; review every finding and maintain controls.
60-84	Improvement Recommended	One or more controls merit attention and prioritised follow-up.
0-59	At Risk	Material weaknesses may be present; timely validation and action are recommended.

21. Result caching and interface refresh

The interface stores the latest structured results so that cards and score summaries can be redrawn without rerunning checks. This enables immediate language changes. A recheck replaces the relevant cached result; a complete recheck refreshes all controls.

22. Offline AI architecture

The local explanatory service uses llama-cpp-python with a local GGUF model based on Qwen2.5 1.5B Instruct. The service provides controlled access to model generation. An optional locally available Ollama runtime may be preferred when configured; otherwise the embedded local model can be used.

> AI limitation
> AI responses may contain errors. Guidance is informational, must be reviewed before changes are applied, and does not autonomously modify Windows settings or remediate vulnerabilities.

23. AI model initialisation

Model initialisation occurs in the background after application startup to reduce contention with initial assessment activity. Availability depends on the selected build and local model configuration. Sufficient local storage is required for the model and associated components.

24. Optional local Ollama integration

When a compatible Ollama runtime and model are locally available, CyberScan may route explanatory requests to that local runtime. This is optional and configuration-dependent. Any external components remain governed by their own configuration and operational controls.

25. Request serialisation and thread safety

Embedded model access is treated as a shared, non-thread-safe resource. A single lock serialises model requests where required for stability, including report explanations and Wiki chat requests. Maintainers must preserve this controlled path and avoid creating unmanaged model instances.

26. Internationalisation

A central English and Italian catalog maps semantic codes to interface text. The backend returns language-neutral codes; the interface resolves them at render time. This separation supports consistent terminology and avoids rescanning after a language change.

27. Language persistence

The selected language is stored locally and restored for later sessions. The active choice is also applied to the Wiki window. Persistence failure should degrade gracefully to an available default without affecting assessment results.

28. Local data processing

Core checks and scan-result processing are designed to run on the Windows workstation. The AI assistant is designed for local operation and does not require a cloud-based AI subscription. Configuration-dependent external tools may have their own behaviour.

29. Privacy and user-controlled sharing

CyberScan follows a local-first processing approach designed to minimise external data transmission. No security report is automatically submitted to CSAEC. The user deliberately initiates report sharing and remains responsible for reviewing its contents and recipients.

> Third-party boundary
> Opening the default email client or using an optional external component may involve third-party software or services. Their privacy, retention, and transmission behaviour is outside CyberScan.

30. Report generation

The report workflow compiles the available control outcomes into a readable summary. A report represents the captured scan state and should include the assessment time, score classification, individual findings, and limitations. Users should verify sensitive information before sharing.

31. Email client integration

Contact actions prepare a message in the user's default email client for review. CyberScan does not automatically send the message. Recipient configuration must be validated for the distributed build, and the user must deliberately complete transmission.

32. Windows packaging and distribution

CyberScan is packaged for Microsoft Windows 10 or later, 64-bit edition. PyInstaller supports the application packaging approach. The exact installer or archive workflow may vary by authorised distribution build and is not prescribed here.

33. Dependency management

Dependency class	Maintenance requirement
Python packages	Pin, review, and test compatible versions for each release.
Assessment components	Validate presence, licensing, signatures, and invocation in the authorised Windows build.
Local AI model	Verify model identity, file integrity, storage needs, and configuration.
Optional runtime	Treat Ollama availability as optional and fail safely when absent.

34. Error handling

Controls should return a clear limited or error result rather than fabricate a healthy state. Expected failure classes include insufficient permissions, unavailable components, timeouts, unreachable services, malformed output, and model initialisation failure.

35. Logging considerations

Operational logging should support diagnosis while minimising collection of sensitive host, network, or report content. Release builds should define appropriate verbosity, retention, access, and redaction. Logs must not be treated as a substitute for security monitoring.

36. Security design principles

- Local-first processing and user-controlled sharing.
- Least necessary exposure between the interface and backend.
- Semantic result contracts and consistent control naming.
- Clear distinction between observation, inference, and confirmed risk.
- Fail-safe presentation when evidence is incomplete.
- No autonomous remediation or hidden external submission.

37. Known limitations

- A completed scan does not prove that the workstation is free from vulnerabilities.
- Results are point-in-time and can be affected by permissions, filtering, segmentation, and component availability.
- Vulnerability checks may produce false positives and false negatives.
- The score is not certification, compliance evidence, or an insurance decision.
- AI output may be incomplete or incorrect and requires review.
- Backup configuration does not prove successful restoration.
- Device discovery is not a complete asset inventory.

38. Troubleshooting

Symptom	Likely area	Recommended action
A control is unavailable	Permissions or missing component	Confirm suitable permissions and the authorised build components; retry.
Few network devices appear	Segmentation, filtering, or inactive hosts	Confirm network context and interpret the result as partial visibility.
AI assistant is not ready	Model initialisation or local runtime	Allow initialisation to complete and verify local configuration.
Report message does not open	Default email client configuration	Confirm that a default email application is configured.
Language text does not refresh	Catalog or persistence issue	Change language again or restart; results need not be rescanned.

39. Maintenance guidelines

- Preserve the stable semantic result contract and translation keys.
- Route all embedded model requests through the serialised AI service.
- Test every control on supported 64-bit Windows builds with standard and restricted permissions.
- Verify dependency availability, error paths, and report recipient configuration before release.
- Render and inspect all user-facing reports and guides after material changes.

40. Extension guidelines

A new control should have a defined purpose, evidence boundary, severity mapping, score contribution, general remediation text, limitations, stable code, and translations. It must not silently broaden network scope or external communication. New assessment components require security, licensing, packaging, and error-handling review.

Extension checklist

Gate	Required evidence
Scope	Authorised target and clear user value
Contract	Stable fields, codes, levels, and score mapping
Safety	No disruptive action; bounded timeouts and safe failure
Privacy	Local processing and sharing implications documented
Quality	Windows test coverage and user-visible limitations

41. Glossary

Term	Definition
Assessment	A point-in-time review of available security indicators.
Control	One of the twelve defined security checks.
EDR	Endpoint detection and response capability provided by a separate security product.
GGUF	A local model file format used by compatible inference engines.
Local-first	Designed to process core information locally and minimise external transmission.
Nmap	A specialised network discovery and service inspection component.
Ollama	An optional locally available runtime for compatible language models.
Wapiti	A web vulnerability assessment component used for selected reachable local services.

Support
For validated implementation, deployment, or finding review, use the authorised CSAEC support channel supplied with the distributed build.