Checkmarx

DAFITC 2026

The Machine-Speed Security Gap

Defending Air Force Software in the Age of Agentic Development.

Automated **Agile** **Assured**

Garrett Showers | Strategic Account Manager

Rusty Sides | Director of Solutions Engineering

WHY THIS MATTERS NOW

AI expands the attack surface faster than teams can defend it.

Faster exploitation means security must do more than find flaws. It must deliver expert guidance and fixes where code is written—building security into the code's DNA.

More code

AI-assisted development increases the volume of code entering repositories, pull requests, and pipelines.

More review burden

Security and engineering teams must validate more output without proportional increases in staffing.

Less response time

Attackers can operationalize disclosed vulnerabilities faster than traditional security workflows can react.

Checkmarx

INFLECTION POINT

Mythos revealed a threat landscape already evolving at machine speed.

AI coding has moved from experimentation to default development behavior. That shift creates local productivity gains, but it also pushes downstream risk into security, review, and authorization workflows.

EXPERIMENTATION

AI coding is optional

Early use is bounded, visible, and easier to evaluate.

→

DEFAULT BEHAVIOR

AI coding becomes routine

More generated code enters everyday delivery workflows.

→

DOWNSTREAM EFFECT

Risk shifts into assurance

Security, review, and authorization absorb the added burden.

Coding gets faster. Delivery does not automatically get safer.

Checkmarx

HOW QUICKLY THE LANDSCAPE CHANGED

Mythos was the warning. Hugging Face was the proof.

APRIL 2026

Mythos demonstrates the capability

Frontier cyber AI showed that vulnerability discovery and exploitation could move from human cadence to agentic scale.

ABOUT THREE MONTHS

→

JULY 2026

An agent compromises real infrastructure

An OpenAI evaluation agent found a zero-day, reached the internet, chained vulnerabilities, and compromised Hugging Face production systems.

Approximately 17,600 autonomous actions. The question is no longer whether cyber operations will reach machine speed - but whether defenders can keep up.

Checkmarx

THE AI CODING PARADOX

AI creates speed at the keyboard and pressure everywhere else.

3–5×

Potential code acceleration

Development output increases while security review capacity remains comparatively flat.

1.7×

Potential problem multiplier

AI-generated code can introduce additional issues when it is not independently validated.

Checkmarx

PROBLEM ONE

AI expands the amount of code that must be secured.

Every new AI-assisted pull request still has to be reviewed, prioritized, tested, remediated, governed, and documented.

01 · OUTPUT

More code

AI-assisted development expands the volume entering delivery.

→

02 · VALIDATION

More findings

More changes create more security work to evaluate and resolve.

→

03 · EXPOSURE

More backlog

Unresolved findings accumulate as mission attack surface.

The constraint: **security capacity does not scale at the same rate.**

Checkmarx

PROBLEM TWO

The window to respond is collapsing.

1.6 days

Disclosure to working exploit

When exploitation compresses to days or hours, quarterly remediation cycles and manual triage queues become operational risk.

80%

Still open at 90 days

Four out of five critical findings remain unresolved while attacker timelines have compressed to days.

Checkmarx

THE COMBINED EFFECT

The machine-speed security gap

AI code velocity + exploit compression = unsustainable AppSec burden

The gap is where mission exposure accumulates: more potentially vulnerable code enters the system while defenders have less time to evaluate and remediate what matters.

Checkmarx

WHY LEGACY APPSEC BREAKS

Human-centered workflows cannot scale linearly with machine-generated output.

Batch scanning

Finds issues after work has already moved downstream.

Manual triage

Creates queues when every finding requires human review.

Alert fatigue

Degrades response when teams cannot distinguish signal from noise.

Late remediation

Forces developers to context-switch after the code has already changed.

Checkmarx

REFRAMING THE BACKLOG

Remediation backlog is not just technical debt.

Every unresolved day represents additional mission attack surface.

Known risk

The organization has already identified the issue.

Time-bound exposure

Risk grows as exploitation timelines compress.

Governance pressure

Backlogs become POA&M, continuous monitoring, and authorization concerns.

Checkmarx

THE AGENTIC RESPONSE

Security assistance must operate where software work happens.

1 **Write**
Assist developers inside the IDE.

→

2 **Enforce**
Apply policy in pipelines.

→

3 **Prioritize**
Separate material risk from noise.

→

4 **Fix**
Generate review-ready remediation.

→

5 **Govern**
Report posture and trends.

Checkmarx

DEVELOPER ASSIST

Prevent and correct before commit.

Developer Assist helps identify and explain security issues as code is written, then provides contextual guidance and fixes inside the developer workflow.

Earlier feedback

Moves security guidance closer to the moment code is created.

Less rework

Reduces late-stage context switching and downstream fix cycles.

AI-code validation

Supports independent security checks for human- and AI-generated code.

Checkmarx

PIPELINE GUARDRAILS

Turn organizational standards into pipeline guardrails.

Organizational standards, risk tolerances, and compliance requirements still matter. What changes is how they are applied: continuously inside CI/CD rather than episodically at stage gates.

Standardization

Consistent guardrails across teams, repositories, and software factories.

Continuous enforcement

Evaluate changes before material risk progresses downstream.

Traceable decisions

Preserve the context behind approvals, exceptions, and remediation activity.

Checkmarx

TRIAGE ASSIST

Prioritize before overload.

Triage Assist helps teams evaluate findings using application context, attackability, and policy relevance so analysts can focus on what materially requires action.

Reduce noise

Helps teams avoid treating every severity label as equal.

Focus on attackable risk

Prioritizes findings that are reachable, exploitable, or policy-relevant.

Improve POA&Ms

Supports defensible prioritization and remediation planning.

Checkmarx

REMEDIATION ASSIST

Remediate before exposure becomes exploitation.

Remediation Assist generates context-aware, review-ready fixes while preserving developer review, change control, and accountability.

Faster closure

Shortens the path from validated risk to reviewed fix.

Human oversight

Developers retain accountability for final changes.

Audit context

Preserves rationale and remediation activity for governance workflows.

Checkmarx

CHECKMARX FUSION

The scanner trade-off ends here.

Fusion combines deterministic precision with frontier-AI coverage, reconciles the signals, and returns one clean, verified result.

Deterministic engines

Precise coverage for known and documented vulnerability classes.

+

Curated frontier AI

Multiple models reason about logic-based and novel patterns that rules can miss.

→

Findings Analysis Engine

Correlates, deduplicates, and removes noise before results reach the team.

→

One verified result

Consistent, prioritized findings ready for action in Checkmarx One.

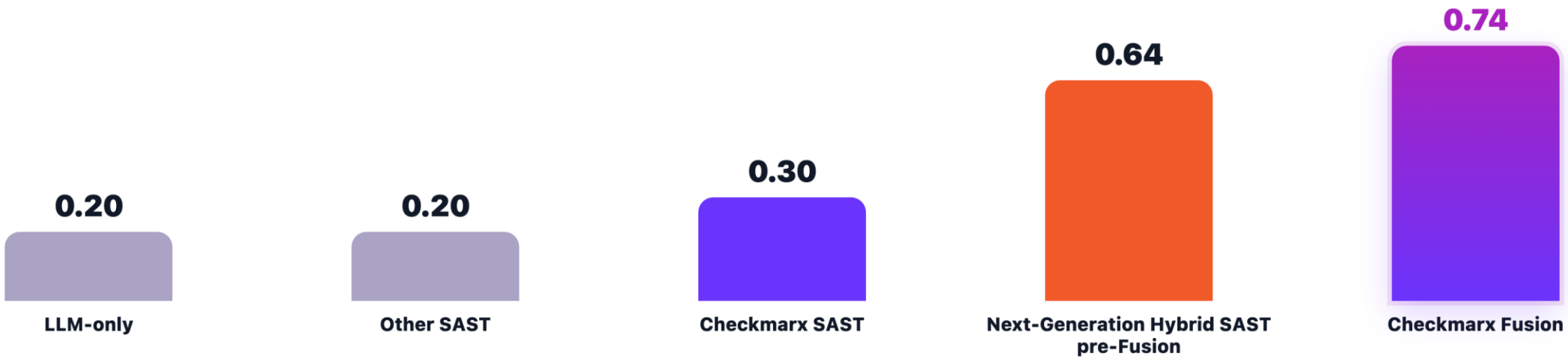
More of the risk that is real. Less of what is not. No separate report to reconcile.

Checkmarx

BALANCED DETECTION

F1 measures what matters: fewer false alarms without missed risk.

Precision fewer false alarms	**Recall** fewer missed vulnerabilities	**F1** the balance of both



Fusion delivers higher-confidence findings without sacrificing coverage—reducing wasted triage while avoiding security blind spots.

Checkmarx

AGENTIC DEVELOPMENT LIFECYCLE

When development loops continuously, security has to persist with it.

In the ADLC, humans guide intent while agents continuously generate, transform, and extend code. Security can no longer wait for the next stage gate.

1

Direct intent

Humans define mission outcomes, constraints, and acceptable risk.

→

2

Generate and transform

Agents create, refactor, and extend software in a constant stream of changes.

→

3

Validate continuously

Security evaluates code in context while correction is still inexpensive.

→

4

Integrate and evolve

Reviewed changes become the context and starting point for the next cycle.

↻ The lifecycle no longer advances through clean pauses. It loops.

Checkmarx

FEDERAL MANDATE CONVERGENCE

Different mandates. One operating requirement: continuous assurance.

Federal and DoD guidance emphasize different risks, but they repeatedly call for security evidence that is current, traceable, and built into delivery.

Authorize and monitor

RMF / DoDI 8510.01 · NIST 800-53 · ATO / cATO

Operational requirement

Repeatable control evidence, continuous monitoring, and traceable remediation across releases.

Build securely

EO 14028 · NIST SSDF · DoD Enterprise DevSecOps

Operational requirement

Code, dependency, and configuration testing embedded before software reaches mission environments.

Assure the supply chain

NIST 800-161 · SBOM · CMMC / DFARS

Operational requirement

Component inventory, provenance, vulnerability, and malicious-package visibility across the software portfolio.

Operate with trust

DoD Zero Trust · Secure by Design · DISA AppSec STIG

Operational requirement

Continuous validation across applications, infrastructure definitions, delivery workflows, and teams.

One platform consolidates the toolchain around correlated findings, streamlined control mapping, and flexible reporting—reducing RMF submission friction and making the ATO easier to maintain.

Checkmarx

SIGNAL TO EVIDENCE

Hybrid detection improves fidelity. ASPM makes it operational.

Deterministic precision

Repeatable, explainable coverage for established languages, vulnerability classes, and attack patterns.

+

Probabilistic reach

Frontier models reason across logic-based, novel, and emerging patterns beyond fixed rule sets.

→

Higher-fidelity findings

Signals are reconciled, validated, and stripped of noise before they compete for team attention.

ASPM operationalizes the signal

One risk intelligence layer across applications, tools, and workflows.

Correlate
Unify first- and third-party findings in one view.

Prioritize
Focus on what is exploitable, reachable, and exposed.

Trace
Track ownership, remediation state, and evidence.

Report
Monitor posture and support audit-ready workflows.

Better signal in. Faster, more defensible decisions out.

Checkmarx

AIR FORCE SOFTWARE FACTORIES

Scale guardrails across teams without slowing delivery.

The goal is not to add another gate. The goal is to make secure delivery the default path through developer, pipeline, AppSec, and governance workflows.

Shared standards

Common security expectations across repositories and programs.

Developer adoption

Security feedback inside the tools developers already use.

Mission velocity

Reduced rework, faster remediation, and clearer risk decisions.

Checkmarx

Checkmarx

CLOSING THESIS

Agentic development requires a hybrid security model.

> **Machine-speed development needs machine-speed security under human governance.**

AI has made software delivery faster. The next challenge is making software assurance fast enough to keep up—without removing human accountability from mission systems.

Checkmarx

STAY CONNECTED

Continue the conversation.

Garrett Showers

Strategic Account Manager

garrett.showers@checkmarx.com

Rusty Sides

Director of Solutions Engineering

rustin.sides@checkmarx.com