

International Journal of **Technology and Systems** (IJTS)

**Event-Driven GitOps for Continuous STIG-Based Compliance on
Regulated Kubernetes Fleets: Measuring Patch Exposure on
Amazon Linux 2023**



**CARI
Journals**

Event-Driven GitOps for Continuous STIG-Based Compliance on Regulated Kubernetes Fleets: Measuring Patch Exposure on Amazon Linux 2023



Pavankumar Koneru,



Sai Kumar Pola

<https://orcid.org/0009-0002-5738-1492>, <https://orcid.org/0009-0003-0014-1559>

Accepted: 15th September, 2026, Received in Revised Form 30th September, 2026, Published: 15th October, 2026

ABSTRACT

Purpose: Federal Kubernetes fleets operating under a FedRAMP High authorization boundary must remain continuously aligned with a Security Technical Implementation Guide (STIG) baseline while also consuming upstream security patches within bounded time. In immutable-infrastructure and GitOps operating models both obligations are discharged by periodic rebuilding of a hardened machine image, yet the rebuild cadence is typically set by organisational convention rather than by the arrival process of the fixes it is meant to consume. This paper measures the exposure and cost consequences of that mismatch for Amazon Linux 2023.

Methodology: Using a dated series of 113 Amazon Linux 2023 security repository releases over a 41-month window, we compare seven trigger regimes — calendar-driven and event-driven — under a phase-averaged exposure model, evaluating both the unpatched exposure window and annualized build cost.

Findings: Repository releases arrive approximately eight times more frequently than the documented quarterly cadence, and an event-driven GitOps trigger dominates every calendar regime evaluated on both exposure and cost. The dominance follows from a structural feature of the arrival process, not from any operational parameter, and is invariant across a twenty-four-fold range of pipeline durations.

Unique contribution to theory, practice and policy: Rebuild cadence is the wrong control variable for continuous compliance in this setting; the relevant variable is whether build issuance is coupled to patch availability. The result gives practitioners and compliance program owners a measured basis for replacing calendar-based rebuild schedules with event-coupled triggers, without incurring the cost penalty of a naive “rebuild more often” strategy.

Keywords: *GitOps, STIG Compliance, FedRAMP High, Amazon Linux 2023, Patch Exposure*

1. INTRODUCTION

Federal workloads that operate under a FedRAMP High authorisation (National Institute of Standards and Technology, 2020) inherit two continuously-binding obligations on their operating-system fleets. The first is configuration compliance against a published STIG baseline. The second is vulnerability currency: the operating system must consume upstream security fixes within bounded time. In immutable-infrastructure architectures, which now describe essentially all production Kubernetes deployments (Morris, 2020), both obligations are discharged by the same mechanism. A hardened machine image is built, validated, and used to launch nodes; when the image is rebuilt, nodes are replaced. Compliance and currency therefore share a single control variable: the rebuild cadence.

That cadence is, in practice, chosen by convention. Quarterly, monthly, and sprint-aligned rebuild schedules are common; the choice is rarely defended against the arrival process of the patches being consumed, and the resulting exposure window is rarely measured. This paper supplies that measurement for Amazon Linux 2023, whose adoption is now widespread and time-pressured as Amazon Linux 2 has reached end of life. A practitioner assembling an AL2023 compliance pipeline today will reasonably design its rebuild trigger around AWS's documented quarterly release cadence; we show that the actual patch arrival rate is nearly an order of magnitude faster, and that an event-driven GitOps trigger reduces the resulting exposure window without incurring the cost of a naive "rebuild more often" strategy.

We investigate two questions. First, how is unpatched exposure distributed across calendar-driven and event-driven rebuild regimes given the empirical arrival process of AL2023 security repository releases? Second, how do build volume and annualised cost vary across those regimes, and does the expected trade-off between exposure and cost obtain? Our measurements are lower bounds on total exposure, as the interval between vulnerability disclosure and vendor fix availability lies outside operator control and is excluded from our clock.

2. RELATED WORK

The window of exposure between vulnerability disclosure and patch availability was formalised by Frei et al. (2006) and quantified across tens of thousands of advisories; we adapt the construct to a later clock (fix availability to fleet deployment) and a different unit (a rebuilt image, not a live host). Zhou et al. (2010) observed that dormant virtual-machine images fall outside conventional patching entirely, and Shu et al. (2017) found persistent, measured staleness across a large sample of Docker Hub images relative to what upstream had already fixed. Neither couples an image's rebuild trigger to the arrival process of a specific distribution's fixes, which is the gap this paper addresses. Angermeir et al. (2024) note that automated continuous security compliance remains comparatively under-researched despite growing industrial adoption; the present paper is one measurement in that space.

The pipeline we evaluate is a GitOps pipeline in the sense defined by Beetz and Harrer (2022): a declarative desired state maintained in Git, an automated reconciler that converges the running system with that state, and every operational change originating as a versioned commit. The GitOps literature has focused primarily on application delivery; the present paper applies the same operating model to the security-image control plane and measures the exposure consequences of coupling the trigger to the upstream arrival process.

Three practitioner references occupy adjacent positions in the AL2023 and Amazon EKS setting. Toh (2022) documents the canonical vendor pattern for STIG-compliant EKS images, targeting Amazon Linux 2 with Packer; it defers scanner verification to the reader and provides neither an automated compliance gate nor a change-driven trigger. Gupta et al. (2026) present a substantially automated GitOps pipeline in which EventBridge polls the EKS AMI parameter twice daily, a language model generates a risk assessment from release notes, and a pull request drives ArgoCD and Karpenter to roll nodes; it advances image currency and applies no hardening, and its approval artifact is a generated narrative rather than a compliance verdict responsive to an authorisation boundary. Hamara et al. (2026) present SHIELDS, a four-agent language-model system remediating up to 73.3% of OpenSCAP findings on Rocky Linux 9; it addresses remediation authoring and is orthogonal to trigger and gate design. None applies a fail-closed compliance gate between patch detection and node advancement, and none measures the exposure consequences of trigger cadence.

3. METHOD

3.1 Exposure Model

For a patch p , the exposure window is the interval during which the fleet runs software known to be vulnerable and for which a fix is consumable:

$$E(p) = t_{\text{deployed}}(p) - t_{\text{available}}(p) = (t_{\text{trigger}} - t_{\text{available}}) + T_{\text{pipeline}}$$

The first term is detection latency, determined entirely by the trigger regime; the second is the deterministic duration of the build, scan, and rolling replacement stages, independent of the trigger.

3.2 Availability Series

We use dated repository releases as the availability series. AWS states that AL2023 security advisories post when the corresponding fix enters the repository, making a repository-release date a defensible proxy for the earliest deployable time of the fixes it carries. Alternatives were rejected on tractability grounds: the ALAS2023 advisory index is a single unpaginated table exceeding retrievable size, the syndication feed is a rolling window rather than a durable archive, and no OVAL vulnerability feed is published for AL2023. We validated the constructed series against an independent sample of 37 individually-read advisory pages spanning the observation window; disagreement in dating was within one day for every sample point. The series was harvested from the thirteen AL2023 quarterly release-notes pages (Amazon Web Services, n.d.), yielding 117

dated releases between 15 March 2023 and 3 August 2026. Four releases flagged by AWS as recalled were excluded as not consumable on their nominal date, leaving $N = 113$ availability events over 1,237 days.

3.3 Trigger Regimes and Phase Averaging

Seven regimes were evaluated: quarterly, monthly, three-weekly, weekly, daily scheduled, event-driven with six-hour evaluation, and event-driven push. For each calendar regime, exposure is sensitive to the arbitrary alignment between the rebuild schedule and the release series; to prevent favourable tuning, every integer phase offset within the period was evaluated and pooled (twenty-one offsets for the three-week regime, ninety for the quarterly regime, and so on), and reported statistics are computed over the union. Polled regimes assume uniform arrival within the poll interval at one-hour granularity.

The pipeline duration T_{pipeline} was set to 4.0 hours based on operator report from a production Amazon EKS environment operating hardened AL2023 node groups with Tenable as scanner of record; the same site reported a three-week rebuild cadence, which we treat as the baseline. Both are practitioner-reported rather than instrumented; Section 5 treats them as threats to external validity, and Section 4.1 reports a sensitivity analysis across pipeline durations from 1 to 24 hours.

3.4 Cost Model

Annualised cost is decomposed into build compute (build and test instances at published `us-east-1` on-demand rates), node-replacement churn during rolling upgrades, AWS Config configuration items generated per replaced node, and a regime-independent fixed component covering Inspector, Config rule evaluations, storage, and continuous integration. We evaluate the model for a fifty-node fleet as a reference size; marginal costs scale linearly and the fixed component is invariant.

3.5 Architecture Under Evaluation

Figure 1 shows the pipeline whose regimes we evaluate. Build execution is conditioned on upstream change through Image Builder's `EXPRESSION_MATCH_AND_DEPENDENCY_UPDATES_AVAILABLE` start condition, so a frequent schedule evaluation issues a build only when either the AL2023 base image or an applied STIG component version has changed. This coupling of trigger to upstream arrival is the mechanism that makes the dominance result of Section 4.2 realisable rather than merely theoretical.

A two-layer fail-closed compliance gate sits between build and node advancement. The validate phase runs OpenSCAP configuration evaluation with severity-graded thresholds so that Category I findings block unconditionally and Category II findings block above a governed threshold; the test phase runs Amazon Inspector or Tenable against a launched instance and blocks on any high-severity finding. Both gates emit persistent evidence artifacts that satisfy the continuous-

monitoring evidence obligations of FedRAMP High. On a clean gate the AMI identifier is written to SSM Parameter Store, committed to the launch template's declarative desired state in Git, and a GitOps reconciler (ArgoCD or Flux) advances the EKS node group. Every state transition is auditable in Git and Image Builder history, with no human intervention between availability event and node replacement.

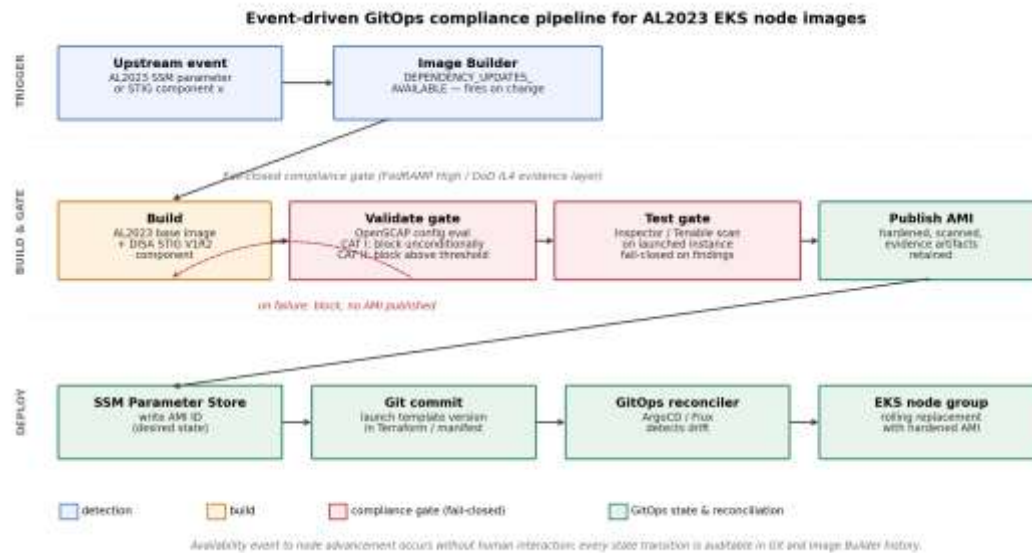


Figure 1: Event-driven GitOps compliance pipeline.

Figure 1. Event-driven GitOps compliance pipeline for AL2023 EKS node images. Trigger events (upstream base-image or STIG-component changes) initiate a build that must pass two fail-closed compliance gates before its AMI is published; the AMI is then committed to a declarative desired state that a GitOps reconciler applies to the running node group.

4. RESULTS

4.1 Exposure

AL2023 repository releases arrive every 11.0 days on average, with median 10.0, standard deviation 5.9, range 1–28, and 90th percentile 21 days. The rate is stable across the observation window with a mild acceleration in 2026 (annual mean inter-release intervals: 10.4 days in 2023, 12.2 in 2024, 11.4 in 2025, and 9.3 in the partial year 2026). Since AWS documents AL2023's minor-version cadence as quarterly, a pipeline designed to the published schedule misapprehends the arrival process by close to an order of magnitude.

Table 1 reports the mean and 95th-percentile exposure window under each regime, together with the reduction relative to the three-week baseline. Figure 2 presents the same data on a log scale.

Table 1. Exposure window by trigger regime ($T_{\text{pipeline}} = 4.0$ h, phase-averaged, $N = 113$).

Regime	Mean (d)	95th pct (d)	Reduction	Factor
Quarterly (90 d)	44.67	85.17	−339.3%	0.2×
Monthly (30 d)	14.67	28.17	−44.3%	0.7×
Three-weekly (21 d), baseline	10.17	19.17	—	—
Weekly (7 d)	3.17	6.17	+68.9%	3.2×
Daily scheduled (24 h)	0.65	1.08	+93.6%	15.7×
Event-driven, 6 h evaluation	0.27	0.38	+97.3%	37.5×
Event-driven, push	0.17	0.17	+98.4%	61.0×

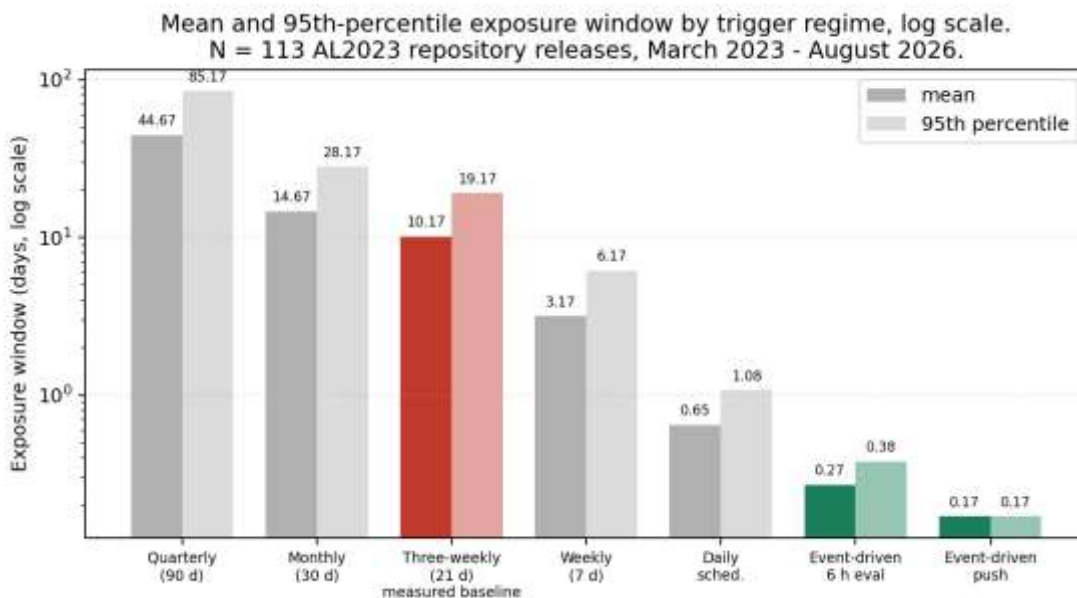


Figure 2: Exposure window by trigger regime.

Figure 2. Mean and 95th-percentile exposure window by trigger regime, log scale. $N = 113$ AL2023 repository releases, March 2023 – August 2026. The three-week regime is the cadence observed at the case site.

The 95th-percentile improvement exceeds the mean improvement for every non-trivial regime and coincides with the mean under event-driven push, because detection latency vanishes and only the deterministic pipeline duration remains. Compliance obligations under FedRAMP High are

typically expressed as time-bounded maxima rather than averages, so the tail is the binding constraint, and it improves more than the mean. A sensitivity analysis across $T_{\text{pipeline}} \in [1, 24]$ hours yields reduction factors ranging from $241\times$ to $11\times$; the *ordering* of regimes is invariant across the entire range, so qualitative conclusions do not depend on the practitioner-reported pipeline duration.

4.2 Build Volume and Cost

Table 2 reports the build count issued by each regime and the fraction of those builds that are *useful*, defined as builds occurring after at least one availability event since the prior build. Calendar regimes fail useful-build efficiency in one of two ways: long periods absorb many releases into one build, while short periods issue builds when nothing has arrived. Event-driven triggering issues exactly N builds by construction, one per availability event.

Table 2. Build volume and useful-build efficiency over 1,237 days.

Regime	Builds	Useful	Efficiency
Quarterly (90 d)	14	14	100%
Monthly (30 d)	42	42	100%
Three-weekly (21 d)	59	58	98%
Weekly (7 d)	177	105	59%
Daily scheduled	1,238	113	9%
Event-driven	113	113	100%

The central finding is that event-driven GitOps triggering does not present a trade-off between cost and latency in this setting; it dominates every calendar regime evaluated on both axes. At 33 builds per year it attains 0.27 days of mean exposure, against daily scheduled rebuilding's 0.65 days at 366 builds per year: 2.4 times lower exposure with 91% fewer builds. No point on the calendar frontier is non-dominated. The mechanism is structural: a calendar decouples build issuance from the quantity it purports to track, and the AL2023 arrival process is irregular enough (standard deviation 5.9 days across a 1–28 day range) that no fixed period tracks it well.

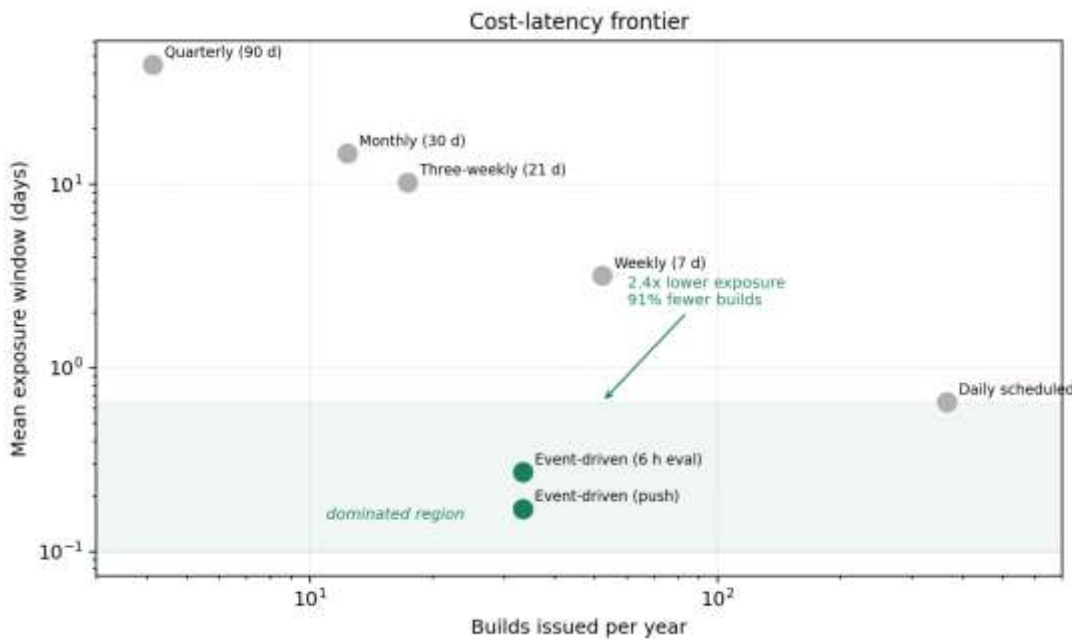


Figure 3: Cost-latency frontier.

Figure 3. Cost-latency frontier. Event-driven configurations (green) attain lower exposure than every calendar regime while issuing fewer builds than daily scheduling. No calendar point is non-dominated on both axes.

Annualised cost for the fifty-node reference fleet at published `us-east-1` rates is \$3,106 for the three-week baseline, \$3,152 for event-driven triggering, and \$4,111 for daily scheduled rebuilding. Cost is dominated by the regime-independent fixed component covering Inspector, Config, storage, and CI/CD; the marginal cost that scales with cadence is node-replacement churn, not build compute. At 366 builds per year the churn term (\$1,024) is thirty-two times the build-compute term (\$32). This is the cost that a naive “rebuild more often” strategy incurs, and event-driven triggering avoids it by rebuilding only when there is a fix to consume.

5. LIMITATIONS

Construct and internal. We use dated repository releases as a proxy for fix availability. AL2023 documentation states that advisories post when the fix enters the repository, and our 37-point advisory sample was consistent with the release series, but the correspondence is not exact. Our measure is time-to-patch, not compromise probability; we do not weight by severity, since the only obtainable severity data was a twelve-day sample skewed by bulk livepatch advisories. The disclosure-to-fix interval is excluded, so reported windows are lower bounds. Four recalled releases were excluded from the availability series; including them changes N by 3.4% and alters no reported ordering. Polled regimes assume uniform arrival within the poll interval, whereas

AWS releases cluster in business hours; the effect is bounded by a few hours at six-hour intervals. Calendar regimes are phase-averaged to prevent favourable tuning.

External and conclusion. The pipeline duration and three-week baseline come from a single operating site and are practitioner-reported rather than instrumented; the sensitivity analysis addresses the former across a twenty-four-fold range, and all regimes are reported so readers may substitute their own baseline. The arrival process is specific to AL2023; other distributions will yield different magnitudes, though the structural argument for coupling does not depend on the specific rate. The model is deterministic given the release series and regime definitions, and reported quantities are exact for the observed series; inferential statistics do not apply. Generalisation to future release behaviour assumes rate stability, which 41 months of data support but cannot guarantee.

6. CONCLUSION

Rebuild cadence is the wrong control variable for continuous STIG compliance of regulated Kubernetes fleets under FedRAMP High authorisation. What matters is whether build issuance is coupled to the arrival process it serves. Once coupled through an event-driven GitOps trigger, frequency ceases to be a design parameter and becomes an output: 113 builds over 41 months at the observed AL2023 rate, because that is how many availability events occurred. The operative engineering question then shifts from how often to rebuild to how quickly a build can complete once triggered. That is a question engineering teams are better equipped to answer, and one whose answer is worth much more, in exposure terms, once the trigger has been coupled to the upstream arrival process.

For practitioners, the actionable next step is to audit existing rebuild schedules against the arrival process of the patches they are meant to consume, and to replace fixed calendar triggers with an event-coupled gate wherever fail-closed compliance evidence obligations apply. For researchers, the open question is whether the same structural dominance holds for other distributions and patch arrival processes, and whether an analogous coupling can be extended to configuration-drift detection beyond the rebuild-time compliance gate evaluated here.

ACKNOWLEDGEMENTS

The authors thank colleagues in DevOps and platform engineering for discussions that shaped the trigger-regime comparison and for feedback on earlier drafts of this work.

REFERENCES

- Amazon Web Services. (n.d.). *Amazon Linux 2023 release notes*. Retrieved August 8, 2026, from <https://docs.aws.amazon.com/linux/al2023/release-notes/relnotes.html>
- Angermeir, F., Fischbach, J., Moyón, F., & Mendez, D. (2024). Towards automated continuous security compliance. In *Proceedings of the 18th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. <https://arxiv.org/abs/2407.21494>
- Beetz, F., & Harrer, S. (2022). GitOps: The evolution of DevOps? *IEEE Software*, 39(4), 70–75. <https://doi.org/10.1109/MS.2021.3119106>
- Frei, S., May, M., Fiedler, U., & Plattner, B. (2006). Large-scale vulnerability analysis. In *Proceedings of the 2006 SIGCOMM Workshop on Large-Scale Attack Defense* (pp. 131–138). <https://doi.org/10.1145/1162666.1162671>
- Gupta, P., Sharma, A., & Tak, A. (2026, May). AI-powered event-driven Amazon EKS AMI updates with GitOps. *AWS Containers Blog*. <https://aws.amazon.com/blogs/containers/ai-powered-event-driven-amazon-eks-ami-updates-with-gitops/>
- Hamara, A., Horne, D., Rojas, A., Kurniawan, T., Lamothe, S., Suresh, V., Turoci, N., & Wong, L. (2026). SHIELDS: Automating OS hardening with iterative multi-agent remediation. In *Proceedings of the International Conference on Security and Management (SAM'26)*. <https://arxiv.org/abs/2606.05476>
- Morris, K. (2020). *Infrastructure as code: Managing servers in the cloud* (2nd ed.). O'Reilly Media.
- National Institute of Standards and Technology. (2020). *Security and privacy controls for information systems and organizations* (NIST Special Publication 800-53, Rev. 5). <https://doi.org/10.6028/NIST.SP.800-53r5>
- Shu, R., Gu, X., & Enck, W. (2017). A study of security vulnerabilities on Docker Hub. In *Proceedings of the Seventh ACM Conference on Data and Application Security and Privacy* (pp. 269–280). <https://doi.org/10.1145/3029806.3029832>
- Toh, W. (2022, July). Building STIG-compliant AMIs for Amazon EKS. *AWS Containers Blog*. <https://aws.amazon.com/blogs/containers/building-stig-compliant-amis-for-amazon-eks/>
- Zhou, W., Ning, P., Zhang, X., Ammons, G., Wang, R., & Bala, V. (2010). Always up-to-date: Scalable offline patching of VM images in a compute cloud. In *Proceedings of the 26th Annual Computer Security Applications Conference* (pp. 377–386). <https://doi.org/10.1145/1920261.1920317>



2026 by the Authors. This Article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>)