What If We Caught SUNBURST in CI/CD? Rewriting the SolarWinds Playbook with AI-Augmented DevSecOps

Aleksandr Krasnov

Why?

- Anatomy of SUNBURST is a masterclass in evasion
- Erosion of Trust and the Rise of "Assume Breach"
  - Trust as a Vulnerability
  - Zero Trust
- AI can help if you use it right
  - traditional SIEM/EDR limitations
  - behavioral anomaly detections
  - threat hunting

SECTOR BRIEFINGS

Don’t forget about NX

Theme	SUNBURST	NX	Takeaway
Trust = New Vulnerability	Trust in a legitimate software vendor is exploited	Trust in a legitimate maintainer’s token and public GitHub profile is exploited	Zero Trust principles should be adopted
Adversarial Tactics: Stealth -> Automation	Manual and patient human operation	Malicious payload used victim’s AI tools to perform data theft	Secure AI tools, deploy AI-powered security to counter AI-driven threats
Dev Pipeline = New Frontier	Infiltrating the build environment is highly effective for wide-scale compromise	CI/CD pipelines and devs’ local machines are primary targets	“Shift Left” is no longer a best practice, it’s a fundamental strategy
Attacks are faster and more destructive	Long-term, low-and-slow cyber-espionage campaign	High-volume, automated blitz that compromised thousands in a matter of hours	AI-powered automation for defense-in-depth

Timeline of SUNBURST

Initial Compromise & Recon

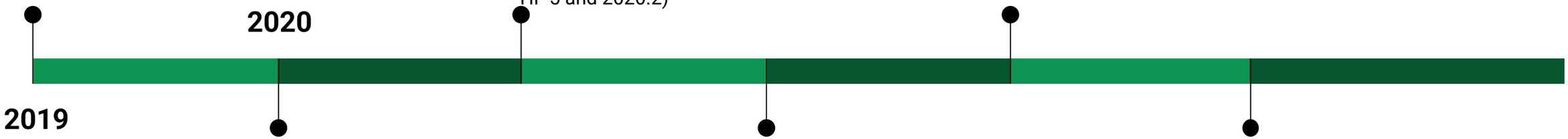
Sep - gained access
Oct - SUNSPOT is injected, “dry run”

Distribution

Mar-Jun - SolarWinds released multiple versions of its Orion platform containing the backdoor (e.g., versions 2019.4 HF 5 and 2020.2)

Post-Exploitation

Mar-Dec - malware is affecting thousands of customers (C2 established via DGA)



Malicious Code Injection

Feb - SUNBURST is injected into `SolarWinds.Orion.Core.BusinessLayer.dll` file

Counter-Forensic

Jun - attackers remove SUNSPOT implant from build servers

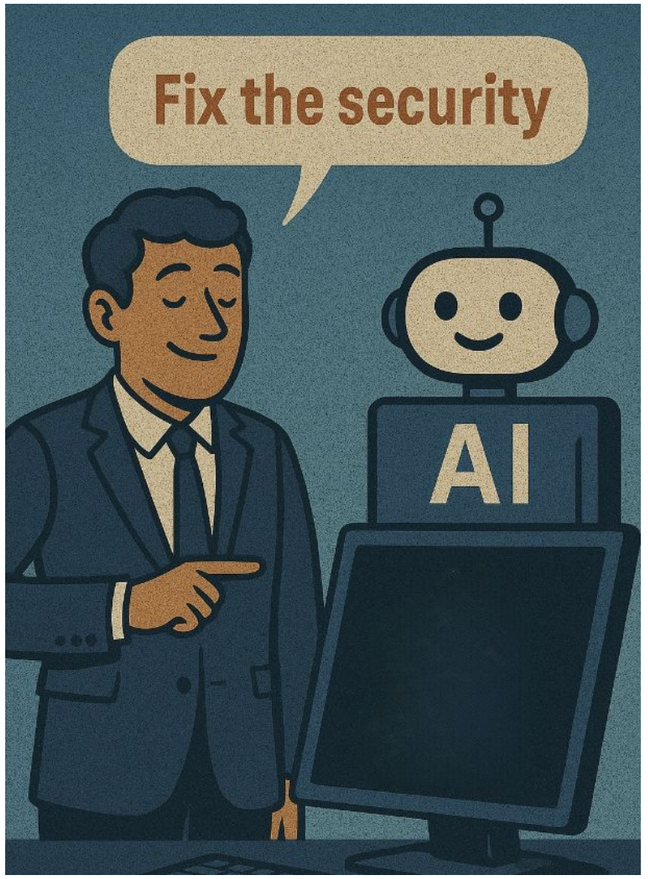
Discovery

Dec - FireEye publicly discloses the attack, coordinated effort of seizure of the main C2 domain `avsvmcloud.com`

Missed Opportunities

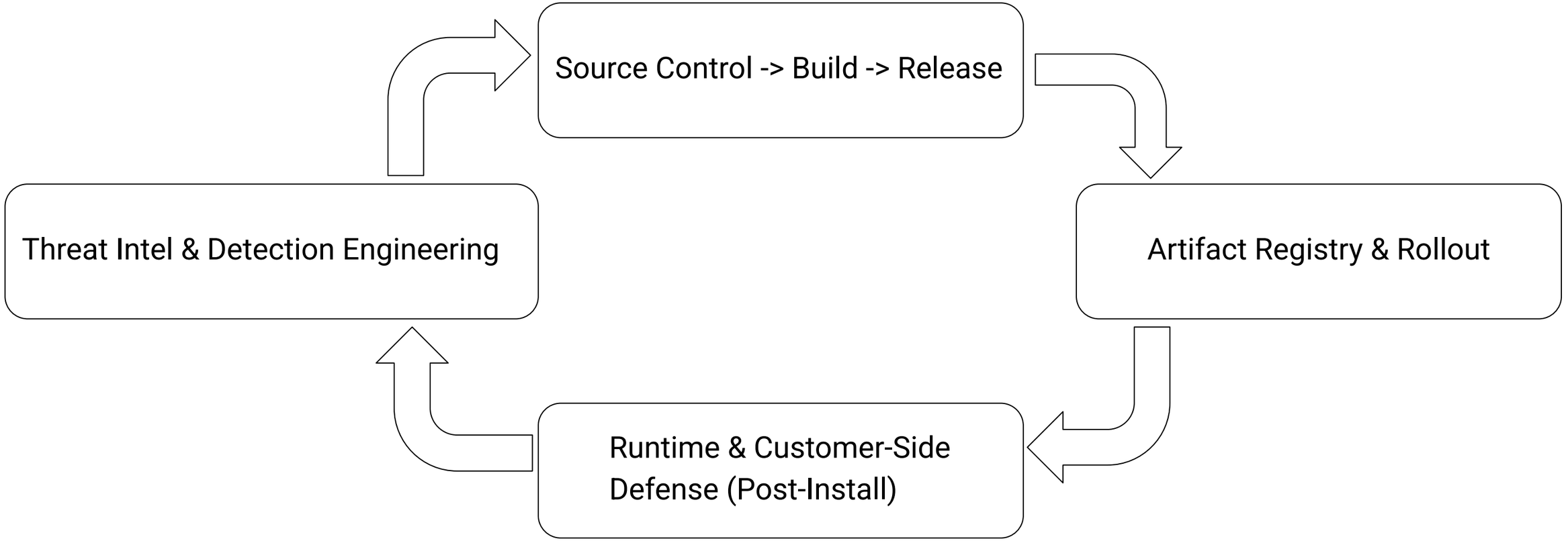
- Build pipeline behavioral anomaly detection
  - File Hash Deviation
  - User/Process Behavior
  - Ephemeral Code Injection
- Code logic and binary analysis for dormant payloads
  - Control Flow Analysis
  - Anti-Forensic Detection
- Network traffic anomaly detection for C2 communication
  - DNS Behavioral Analysis
  - DGA Detection
  - Traffic Mimicry Anomaly
- Continuous Security Posture Management
  - Predictive Threat Intelligence
  - Automated Threat Hunting

Rethinking DevSecOps



?

AI-Augmented DevSecOps



Source Control -> Build -> Release

“Golden-build” anomaly detection

Catch a SUNSPOT-style implant that swaps code during compile

- ❏ process graph builds (compiler/linker ancestry, unexpected msbuild or script invocations)
- ❏ ephemeral file hashes in build workspace
- ❏ build DAG changes
- ❏ timestamps/skews
- ❏ unsigned tools loaded in builders

- ❏ unsupervised anomaly detection per repo/pipeline (autoencoders or Isolation Forest) trained on weeks of known-good builds
- ❏ per-job feature vectors compare current build vs. golden profile

if anomaly_score > threshold, then quarantine artifact, page release manager, and force independent rebuild

SUNSPOT persisted on the build server and silently injected the SUNBURST DLL during compilation

Source Control -> Build -> Release

AI guardrails on provenance & attestations (SLSA/SSDF)

Enforce tamper-evident builds

- ❏ In-toto/SLSA provenance blobs
- ❏ signer identities
- ❏ hermeticity flags
- ❏ builder IDs
- ❏ reproducibility checksums

- ❏ LLM “policy checker” that reasons over the attestation plus pipeline logs (RAG over your controls policy) to explain which SLSA or SSDF requirements are missing and block release if any are critical

Require green checks for ephemeral builders, two-person release, artifact provenance attached, hash-pinned deps, and reproducible build match

SUNBURST shipped digitally signed malware because SolarWinds’ release process trusted the build pipeline without independent provenance checks

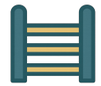
Source Control -> Build -> Release

ML binary-to-source consistency checks

Detect inserted code that isn't in the reviewed source

- ❏ binary CFG and symbol graph vs expected compiler output from source
- ❏ opcode n-grams
- ❏ embedded strings

- ❏ siamese/contrastive models trained to align AST features (from source) with CFG features (from built binary) <- *research prototype/promising direction*
- ❏ Alarms on unexpected diffs (e.g., new network/DNS routines)



Block publication if deltas exceed risk budget, auto-open a PR with the diff summary for human review

SUNBURST's payload was inserted at build time

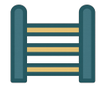
Source Control -> Build -> Release

AI static/dynamic “time-bomb” & obfuscation hunting

Catch delayed-activation backdoors and obfuscation that evaded QA

- ❏ sleep/jitter patterns
- ❏ randomized timers
- ❏ environment/username/AD-domain collection
- ❏ string-construction of hostnames
- ❏ domain-like entropy

- ❏ ML classifiers over IL/bytecode and sandbox traces
- ❏ LLMs summarize sandbox behavior and flag stealth patterns



Quarantine artifacts showing long randomized sleeps or DGA-like DNS routines

Backdoor used delayed activation, environment checks and obfuscation to evade QA

Source Control -> Build -> Release

AI in PR review

Stop suspicious diffs before they hit the build

- ❏ new network code
- ❏ crypto helpers
- ❏ base64 blobs
- ❏ reflection/dynamic Assembly.Load
- ❏ unsigned third-party tooling

- ❏ AST-aware LLM reviewer that produces explainable risk notes and asks authors to justify each risky change
- ❏ Blocks merges without justification

Flagged PRs must include security justification, merge is blocked until AppSec approve or changes are removed

Catching risky primitives before they ever land prevents the conditions that make a build hijack profitable and reduces the attack surface for pipeline implants

Source Control -> Build -> Release

Dependency/third-party risk with graph ML

Spot abnormal SBOM edges and risky dependency updates

- ❏ SBOM graph
- ❏ version drift
- ❏ maintainer trust
- ❏ typosquat heuristics
- ❏ release cadences
- ❏ CVE velocity

- ❏ graph embeddings or GNNs to score “odd” package additions
- ❏ auto-cross check with threat intel and supplier posture (SSDF-aligned supplier requirements)

If graph embeddings flag a high-risk or anomalous package, the pipeline blocks promotion until AppSec performs manual verification

Supply chain attacks often come through package ecosystems (typosquats, compromised maintainers, sudden release cadence changes)

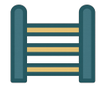
Artifact Registry & Rollout

Attestation-aware progressive delivery

Ensure only attested artifacts roll out

- ❏ signature validity
- ❏ provenance shard
- ❏ model risk scores

- ❏ policy model that fuses scores (anomaly, binary-source diff, static/dynamic risk) into a single release readiness metric



Canary to a telemetry tenant first; auto-revert on elevated beaconing/DNS anomalies

SUNBURST rode on trusted signatures, a signed artifact was trusted by default

Artifact Registry & Rollout

AI-assisted binary diffing between releases

Detect stealthy control-flow or string changes in “minor” hotfixes

- ❑ code flow deltas
- ❑ string/constant changes
- ❑ entropy metrics
- ❑ semantic embeddings
- ❑ import table changes

- ❑ embedding-based diff across functions
- ❑ LLM produces human-readable “what changed and why it’s risky” report

Block promotion if high-risk function deltas aren’t references in PRs or release notes

Malicious DLL evolved across builds, embedding-based diffs surface semantic changes that simple hashes miss

Runtime & Customer-Side Defenses

DNS/DGA anomaly detection

Catch SUNBURST-style C2 early

- ❑ NXDomain burstiness
- ❑ domain entropy
- ❑ subdomain length
- ❑ lexical patterns mimicking cloud APIs
- ❑ per-host rare destinations

- ❑ unsupervised detectors on DNS logs plus a small supervised DGA classifier
- ❑ rule overlays for `*.avsvmcloud.com` and similar patterns

Triage/contain host if score > threshold; enrich with Orion process metadata

Post-install, SUNBURST used rare, high-entropy subdomains under a legitimate-looking domain and NXDomain probing before tasking

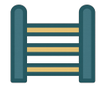
Runtime & Customer-Side Defenses

Host EDR sequence models

Detect orion process doing non-Orion things

- ❑ process ancestry
- ❑ module loads
- ❑ unusual network libraries
- ❑ credential access attempts
- ❑ unsigned DLL injection

- ❑ LSTM/Transformer on syscall/event sequences per application profile
- ❑ Alert on deviations



Auto-isolate endpoint; capture memory for forensics; block egress to suspicious ASN

Orion process began doing non-Orion things (credential access, new libraries, outbound beacons)

Runtime & Customer-Side Defenses

Identity & Lateral movement graph analytics

Limit blast radius even if initial beacon succeeds

- ❑ auth graph (accounts -> groups -> resources)
- ❑ role anomalies
- ❑ service account over-privilege

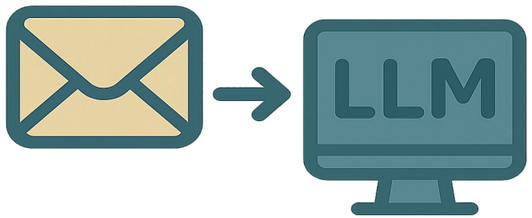
- ❑ graph outlier detection to find impossible trust paths
- ❑ propose just-in-time privilege and segmentation changes

Auto-enforce JIT privilege and set-up MFA for sensitive role changes, quarantine anomalous trust edges pending human approval, block automated Kerberoast-prone SPN changes

After foothold, the campaign relied on privilege escalation and lateral movement. Graph outlier detection exposes impossible trust paths and risky over-privilege before/during abuse

Threat Intel & Detection Engineering

IoC -> Detections



Hunt Playbooks

Posture Gap Analysis

How these controls map to SUNBURST

Build hijack via SUNSPOT	→	Build anomaly detection & reproducible/attested builds
Signed trojanized updates shipped to customers	→	Implement attestation gates + binary-to-source checks + progressive release
Stealthy C2 using DGA subdomains, delayed activation	→	Perform DNS/DGA analytics + behavior sandboxes that look for time-bomb logic
Framework alignment	→	Adopt SLSA for artifact integrity and NIST SSDF for organization-wise secure SDLC practices

SECTOR BRIEFINGS

RACI

Control Area	Specific Measure	Responsible	Accountable	Consulted	Informed
Source Control	- AI/LLM PR review on risky code - Commit signing & provenance enforcement	- Eng Team Leads - AppSec Eng	- Head of Eng - Sec Eng Lead	- AppSec - Devs	- CISO - CISO + Audit
Build Pipeline	- Build anomaly detection - Reproducible builds & independent rebuilds	- CI/CD Ops - Release Eng	- DevSecOps - Release Mgr	- AppSec - DevSecOps	- CISO + Eng Leads - Eng Teams
Artifact Release	- SLSA/SSDF attestation checks - Multi-party release signing	- DevSecOps - Release Eng	- CISO - CTO	- Release Eng + AppSec - AppSec + Legal	- Execs - Security Committee
Testing/QA	- ML-behavioral sandbox - Binary-to-source diffing	- QA Automation - AppSec	- QA Lead - Sec Eng Lead	- AppSec - Release Eng	- Eng Leads - CISO
Deploy/Ops	- AI-fused risk score gating - Progressive rollout w/ telemetry	- DevSecOps - SRE	- SRE Lead - SRE Lead	- AppSec - DevSecOps	- CTO - CISO + Eng Teams
Runtime	- DNS/DGA anomaly detection - Host EDR ML sequence models	- SOC analysts - SOC + endpoint	- SOC Mgr - SOC Mgr	- Network Ops - IT Ops	- CISO + Risk - CISO
Governance	- LLM SSDF/SLSA gap analysis - Red-team pipeline simulations	- CISO - Security Assurance	- CISO - CISO	- DevSecOps + Compliance - Eng + CI/CD teams	- Board - Execs + Audit

POC

CI Logs / SBOMs / DNS / EDR / Auth

Secure Data Lake + Feature Store

Model Train

- Build anomaly (unsup)
- Provenance LLM (policy)
- Binary↔Source (Siamese)
- Sandbox/Obfuscation (seq cls)
- Dependency GNN
- DNS/DGA + EDR seq models
- Identity graph outliers

Model Registry

- Versioned, attested, promoted w/ approval

Serving & Policy

- Risk Fusion Engine
- OPA Release Gate
- Canary/rollback

CI/CD Decisions + SOC Alerts

Core Principles:

- ZT between planes
- Signed & attested: data -> features -> models -> inferences
- Hermetic builds, no default egress
- Human-in-the-loop on all block decisions

SLOs:

- 100% artifacts signed & attested
- Detect DNS/DGA beacons < 15 mins
- Isolate EDR anomalies < 60 mins
- FP block rate < 0.5%

SecTor Sound Bytes

`Signed != Safe`. Attackers can weaponize trust itself

DevSec Ops → AI-Augmented ThreatOps

CI/CD pipeline = attack surface. Treat it like a battlefield

If you don’t hunt threats in your builds, your adversaries will

Q&A