pellera

THREAT INTEL REPORT 2025

Prepared by: Pellera Threat Intel Team
pellera.com | 800.747.8585

SEPTEMBER

Driving Momentum. Accelerating Change. Empowering IT Transformation.

Pellera Technologies was born out of the combined expertise of Converge Technology Solutions and Mainline Information Systems, two industry leaders with over 35+ years of experience and a shared vision for innovation. Together, we empower businesses to achieve greater efficiency, adaptability, and growth for today and tomorrow.

Our commitment is to reshape what's possible with IT, offering advanced solutions in digital infrastructure, cloud, cybersecurity, and AI. We don't just deliver technology—we partner with you to build tailored strategies designed to simplify complexities, unlock opportunities, and drive transformational outcomes.

At Pellera, momentum builds here through collaborative, people-first technology designed to fuel progress and deliver measurable impact.

pellera

CONVERGE TECHNOLOGY SOLUTIONS + Mainline®

Observations for September 2025

September 2025 marked a turning point in cyber warfare. For the first time, threat actors simultaneously breached the three pillars of modern enterprise infrastructure: software development, artificial intelligence, and firmware security. What makes these attacks particularly dangerous is their targeting of trust relationships that organizations rely on daily.

A self-replicating worm called Shai-**Hulud infected over 500 NPM packages**, including CrowdStrike's own repositories. This automated attack harvested GitHub, AWS, and Azure credentials from development environments across 40+ countries, creating cascading risks to production systems. Unlike previous supply chain attacks requiring manual intervention, Shai-Hulud spreads itself automatically through compromised maintainer credentials.

Google Vertex AI and Microsoft Azure AI platforms fell victim to a novel attack exploiting deleted model namespaces on Hugging Face. Security researchers demonstrated how attackers can hijack trusted AI models by registering abandoned namespaces, embedding reverse shells that execute when enterprises deploy what they believe are legitimate models. Thousands of repositories now contain vulnerable references.

HybridPetya ransomware achieved what many thought impossible: bypassing UEFI Secure Boot protections to establish firmware-level persistence. This fourth-generation bootkit survives complete operating system reinstallation, yet unlike the destructive NotPetya, maintains decryption capabilities for ransom collection rather than pure destruction.

The coordinated timing of these breaches suggests a strategic shift in threat actor operations. No longer content with single-vector attacks, adversaries now target multiple foundational technologies simultaneously, ensuring organizational compromise regardless of defensive measures. MSP clients face an unprecedented threat landscape requiring immediate attention across development, AI, and firmware security domains.

Executive Overview

NPM SUPPLY CHAIN COMPROMISE - "SHAI-HULUD"

Audience

- CISO
- IT Operations Managers & Teams
- Risk Management Professionals
- Security Operations Team
- IT Security Managers
- Threat Intelligence Analysts
- Software Development Teams

RISK

GEOGRAPHIC SCOPE

Development environment compromise, credential theft, potential production system access through stolen cloud service tokens

BUSINESS IMPACT

The Shai-Hulud campaign represents the most sophisticated NPM supply chain attack recorded, compromising over 500 packages through automated worm-like propagation mechanisms. The attack initiated with targeted phishing campaigns spoofing npmjs.help, directing package maintainers to credential harvesting pages designed to capture NPM publishing tokens. Once credentials were obtained, attackers deployed a self-replicating payload bundled as bundle.js and disguised as a "System Info App."

The malicious code executes comprehensive credential harvesting operations targeting GitHub personal access tokens, NPM tokens, AWS access keys, GCP service account keys, and Azure credentials. The payload employs TruffleHog for systematic secret scanning across filesystem locations and probes cloud metadata endpoints for additional credential discovery. The attack demonstrates advanced automation capabilities, automatically identifying packages owned by compromised maintainers through npm registry queries, injecting malicious postinstall hooks, bumping patch versions, and republishing infected packages using stolen credentials.

READ MORE: NPM SUPPLY CHAIN COMPROMISE

UEFI RANSOMWARE EVOLUTION - HYBRIDPETYA

Audience

- CISO
- IT Operations Managers & Teams
- Risk Management Professionals
- Incident Response Teams
- Security Operations Team
- IT Managers
- System Administrators

RISK

Targeting Modern UEFI-Based Systems

GLOBAL

GEOGRAPHIC SCOPE

Complete system encryption with firmware-level persistence, potential for widespread organizational disruption similar to NotPetya

INDUSTRY IMPACT

HybridPetya represents a concerning evolution in ransomware capabilities, combining proven Petya/NotPetya architecture with advanced UEFI Secure Boot bypass techniques. The ransomware exploits CVE-2024-7344 to circumvent UEFI Secure Boot protections on unpatched systems, installing malicious EFI applications directly onto the EFI System Partition for firmware-level persistence that survives operating system reinstallation.

The attack methodology involves systematic replacement of legitimate Windows bootloaders with malicious variants that encrypt the Master File Table (MFT) on NTFS partitions using Salsa20 encryption algorithms with 32-byte keys and 8-byte nonces.

Technical analysis reveals HybridPetya employs a specially crafted "cloak.dat" file loaded through a vulnerable but Microsoft-signed "reloader.efi" application to achieve Secure Boot bypass functionality.

READ MORE: UEFI RANSOMWARE EVOLUTION - HYBRIDPETYA

AI MODEL SUPPLY CHAIN ATTACKS - NAMESPACE REUSE

Audience

- **CISO**
- **IT Operations Managers & Teams**
- **Risk Management Professionals**
- **Security Operations Team**
- **AI/ML Development Teams**
- **Cloud Security Teams**
- **Software Development Teams**

HIGH	Affecting Google, Microsoft, and Thousands of Open-Source Repositories GLOBAL	Unauthorized model deployment, potential remote code execution, infrastructure compromise through AI platform access
RISK	**GEOGRAPHIC SCOPE**	**INDUSTRY IMPACT**

Model namespace reuse attacks represent a critical new vector targeting AI model supply chains through systematic exploitation of deleted or transferred namespaces on platforms like Hugging Face. Attackers register previously deleted organization namespaces and upload malicious models with identical names to legitimate models, targeting automated deployment pipelines that reference models solely by namespace identifiers without authenticity verification.

Security researchers successfully demonstrated attack feasibility against Google Vertex AI Model Garden and Microsoft Azure AI Foundry, embedding reverse shell code within hijacked model namespaces to achieve infrastructure access and obtain Azure endpoint permissions. Comprehensive analysis reveals thousands of open-source repositories contain vulnerable model references embedded in code, creating extensive transitive supply chain risks across software ecosystems.

READ MORE: AI MODEL SUPPLY CHAIN ATTACKS

NPM SUPPLY CHAIN COMPROMISE - "SHAI-HULUD" CAMPAIGN

Overview & Impact

The Shai-Hulud NPM supply chain attack represents a paradigm shift in software supply chain compromises, demonstrating automated worm-like propagation capabilities that systematically harvest credentials and propagate through package dependency networks. The attack specifically targets Linux and macOS development environments, activating during npm install operations via postinstall script execution. Critical packages affected include @ctrl/tinycolor with 2.2 million weekly downloads, along with packages from @nativescript-community, @operato, and @things-factory namespaces.

The attack's sophistication extends beyond traditional package injection techniques, implementing comprehensive automation for credential harvesting, package modification, and republishing operations. Attackers created public repositories named "Shai-Hulud" for data exfiltration and attempted to inject GitHub Actions workflows for persistent access. The payload uses multiple base64 encoding layers and webpack bundling to obfuscate its 3.6MB malicious component while maintaining compatibility across development environments.

- Development environment compromise affecting 40+ countries globally
- Systematic credential theft targeting GitHub, NPM, AWS, GCP, and Azure tokens
- Cascading supply chain effects where single compromised maintainer leads to hundreds of infected packages
- Potential production system access through compromised development credentials
- Mobile application risk through shared JavaScript dependencies in React Native, Ionic, and Cordova frameworks

Threat Implications

- Supply chain attacks now demonstrate self-propagating capabilities reducing operational overhead
- Development environment compromise creates direct pathways to production system access
- Credential harvesting enables lateral movement across cloud infrastructure and services
- Automated attack scaling fundamentally changes supply chain threat assessment models

Notable Campaigns

- @ctrl/tinycolor compromise affecting 2.2 million weekly downloads
- CrowdStrike package compromises discovered in attack scope
- Over 500 packages compromised with active propagation contained as of September 18, 2025
- Multiple namespace targeting including @nativescript-community, @operato, @things-factory
- Cryptocurrency transaction interception capabilities targeting web environments
- Mobile application impact through JavaScript dependency sharing across North America and Europe

MITRE ATT&CK Framework Mapping

- **T1195.002 - Supply Chain Compromise:** Software Supply Chain
- **T1078.004 - Valid Accounts:** Cloud Accounts
- **T1552.001 - Unsecured Credentials:** Credentials In Files
- **T1555.003 - Credentials from Password Stores:** Credentials from Web Browsers
- **T1567.002 - Exfiltration Over Web Service:** Exfiltration to Cloud Storage

RETURN >>>

Observations

- Postinstall script execution provides ideal vector for automated credential harvesting during package installation
- npm registry API access enables systematic identification of maintainer-owned packages for targeted infection
- Development environments lack sufficient monitoring for package installation and credential access activities
- GitHub Actions workflow injection provides persistent access mechanism across compromised repositories
- Webhook.site usage demonstrates preference for legitimate services to avoid detection
- Multiple base64 encoding layers effectively obfuscate malicious payloads from static analysis
- TruffleHog employment indicates sophisticated secret detection capabilities targeting multiple credential types

Guidance

Strategic Intelligence

- **Trend**
  - Supply chain attacks evolving from manual injections to automated self-propagating mechanisms
  - Threat actors investing in sophisticated automation technologies for scaled compromise operations
  - Development environment targeting increasing as pathway to production system access
  - Credential harvesting prioritized over immediate monetization for strategic positioning
- **Business Risk Context**
  - Single compromised developer can lead to hundreds of infected packages affecting millions of users
  - Development credential theft creates direct pathways to cloud infrastructure and production systems
  - Supply chain security failures cascade across organizational technology stacks
  - Traditional security controls insufficient for automated supply chain attack detection

Operational Intelligence

- **Threat Vectors**
  - Phishing campaigns targeting package maintainers with credential harvesting pages
  - Postinstall script execution during package installation providing automation opportunities
  - npm registry API exploitation for systematic package identification and modification
  - GitHub Actions workflow injection for persistent access across compromised repositories
  - Cloud metadata endpoint probing for additional credential discovery
- **Monitoring & Detection Gaps**
  - npm package installation processes lack comprehensive security monitoring
  - Postinstall script execution insufficient visibility in development environments
  - npm registry API calls not monitored for bulk modification operations
  - Repository visibility changes and GitHub Actions modifications undetected
  - Cloud metadata endpoint access from development systems not flagged

RETURN >>>

- **Response Actions**
  - Immediate credential rotation for GitHub, NPM, AWS, GCP, and Azure tokens
  - Development environment audit for compromised packages and unauthorized repository changes
  - npm proxy server deployment with package scanning capabilities
  - CI/CD pipeline hardening to prevent compromised development credentials affecting production
  - Enhanced monitoring implementation for package installation and credential access activities

Tactical Intelligence

- **Mitigation Strategies**
  - Disable postinstall scripts globally: npm config set ignore-scripts true
  - Implement npm proxy servers with comprehensive package scanning
  - Deploy package-lock.json with exact version pinning for dependency management
  - Enable npm audit automation in CI/CD pipeline stages
  - Monitor GitHub Actions for unauthorized workflow injection attempts
- **Preventive Measures**
  - Package installation monitoring with postinstall script execution tracking
  - npm registry API call monitoring for bulk modification detection
  - Repository visibility change detection and alerting
  - Cloud metadata endpoint access monitoring from development systems
  - Systematic credential scanning in development environment monitoring
- **Detection Commands**
  - grep -R 'checkethereumw' node_modules/ (scan for specific malware indicators)
  - grep -R 'webhook.site' node_modules/ (detect exfiltration endpoints)
  - npm audit --audit-level high (comprehensive package vulnerability assessment)
  - find . -name "bundle.js" -exec ls -la {} \; (identify suspicious bundled files)

Development Environment Credential Harvesting

Hypothesis: Attackers use compromised npm packages to systematically harvest development credentials from local filesystems and environment variables during package installation processes.

Investigation Steps

1. Monitor npm install operations for unusual postinstall script execution patterns
2. Search for TruffleHog execution or similar secret scanning tools in development environments
3. Detect unauthorized access to .npmrc, .gitconfig, and cloud credential files during package installation
4. Identify processes probing cloud metadata endpoints (169.254.169.254) from development systems
5. Correlate package installation timing with credential file access and modification events

Package Dependency Chain Compromise

Hypothesis: Threat actors exploit compromised maintainer credentials to inject malicious code into multiple packages owned by the same maintainer, creating cascading supply chain effects.

Investigation Steps

1. Correlate npm package updates from single maintainers affecting multiple packages simultaneously
2. Monitor npm registry API calls for bulk package modification operations
3. Detect version bumps across multiple packages from same maintainer within short time windows
4. Identify packages with recent postinstall hook additions or modifications
5. Track maintainer account activity for unusual publishing patterns and credential usage

Repository Manipulation and Data Exfiltration

Hypothesis: Compromised development environments automatically create public repositories for data exfiltration and maintain persistent access through GitHub Actions workflow injection.

Investigation Steps

1. Monitor for unexpected public repository creation with "Shai-Hulud" or similar naming patterns
2. Detect unauthorized GitHub Actions workflow files in existing repositories
3. Identify automatic repository visibility changes from private to public
4. Track unusual repository creation patterns following package installation activities
5. Analyze webhook.site or similar exfiltration endpoint usage from development environments

Sources

- **TrueSec:** 500 NPM Packages Compromised in Ongoing Supply Chain Attack Shai-Hulud
- **Arctic Wolf:** Wormable Malware Causing Supply Chain Compromise of NPM Code Packages
- **Unit 42:** NPM Supply Chain Attack
- **OX Security:** NPM 2.0 Hack - 40 NPM Packages Hit in Major Supply Chain Attack
- **JFrog:** Shai-Hulud NPM Supply Chain Attack - New Compromised Packages Detected
- **StepSecurity:** @ctrl/tinycolor and 40 NPM Packages Compromised
- **NowSecure:** Major NPM Supply Chain Attack - Potential Impact on Mobile Applications
- **Security Alliance:** 2025-09 NPM Supply Chain Attack

UEFI RANSOMWARE EVOLUTION - HYBRIDPETYA

Overview & Impact

HybridPetya demonstrates the evolution of ransomware capabilities to target firmware-level protections, exploiting CVE-2024-7344 to bypass UEFI Secure Boot and achieve persistent infection that survives operating system reinstallation. The ransomware encrypts Master File Table (MFT) structures using Salsa20 encryption while displaying fake CHKDSK messages to avoid detection. Unlike the destructive NotPetya variant, HybridPetya maintains decryption capabilities, indicating threat actors prioritize financial gain over pure disruption.

Current ESET telemetry shows no active spreading campaigns, suggesting the malware remains in proof-of-concept status. However, the demonstrated capabilities represent significant advancement in firmware-level attack techniques. The ransomware creates comprehensive tracking files on the EFI System Partition and generates victim-specific installation keys for targeted deployment. Analysis indicates this represents the fourth publicly documented UEFI bootkit capable of bypassing Secure Boot protections.

Impact

- Complete system encryption with firmware-level persistence surviving OS reinstallation
- UEFI Secure Boot bypass capabilities on unpatched systems exploiting CVE-2024-7344
- Potential for widespread organizational disruption similar to NotPetya's $10 billion global impact
- Advanced persistent threat capabilities challenging traditional backup and recovery strategies
- Fundamental compromise of boot integrity and system trust mechanisms

Threat Implications

- Ransomware evolution now targets firmware-level protections requiring enhanced defensive strategies
- UEFI bypass capabilities represent significant advancement in persistent threat techniques
- Financial motivation prioritized over destructive impact indicating targeted attack potential
- Firmware-level persistence fundamentally challenges traditional incident response and recovery procedures

Notable Campaigns

- No active spreading campaigns detected in current ESET telemetry
- Proof-of-concept demonstration targeting modern UEFI-based systems
- Poland-origin sample uploaded to VirusTotal in February 2025
- Fourth publicly known UEFI bootkit with Secure Boot bypass capabilities

MITRE ATT&CK Framework Mapping

- **T1542.001 - Pre-OS Boot:** System Firmware
- **T1014 - Rootkit:** UEFI Rootkit
- **T1486 - Data Encrypted for Impact**
- **T1490 - Inhibit System Recovery**
- **T1055.012 - Process Injection:** Process Hollowing

RETURN >>>

Observations

- CVE-2024-7344 provides reliable Secure Boot bypass mechanism through signed Microsoft applications
- EFI System Partition modification enables persistent firmware-level access across OS reinstallation
- Salsa20 encryption with 32-byte keys provides strong encryption while maintaining decryption capabilities
- Fake CHKDSK messages effectively delay detection during encryption operations
- Victim-specific installation keys suggest targeted deployment rather than widespread distribution
- Backup bootloader creation indicates sophisticated persistence and recovery evasion techniques
- Microsoft-signed "reloader.efi" exploitation demonstrates supply chain trust relationship abuse

Guidance

Strategic Intelligence

- **Trend**
  - Ransomware operators investing in firmware-level attack capabilities for enhanced persistence
  - UEFI bootkit development indicating widespread research investment in firmware attack techniques
  - Financial motivation prioritized over destructive impact reducing collateral damage potential
  - Proof-of-concept development suggesting future operational deployment preparation
- **Business Risk Context**
  - Firmware-level persistence challenges fundamental assumptions about backup and recovery effectiveness
  - UEFI attack capabilities require specialized incident response and forensic capabilities
  - Traditional endpoint protection insufficient for firmware-level threat detection
  - Critical infrastructure and high-value targets face elevated risk from persistent firmware compromise

Operational Intelligence

- **Threat Vectors**
  - CVE-2024-7344 exploitation through specially crafted "cloak.dat" files
  - Vulnerable Microsoft-signed "reloader.efi" application abuse for Secure Boot bypass
  - EFI System Partition modification for persistent malicious application installation
  - Legitimate bootloader replacement with malicious variants
  - Social engineering during fake CHKDSK operations to delay detection
- **Monitoring & Detection Gaps**
  - UEFI firmware modification detection insufficient in current security monitoring
  - EFI System Partition integrity checking not implemented in standard endpoint protection
  - Bootloader verification and integrity monitoring gaps in enterprise environments
  - CVE-2024-7344 exploitation indicators not monitored in firmware security controls
  - Fake system maintenance message detection not included in user awareness training
- **Response Actions**
  - Immediate UEFI firmware updates addressing CVE-2024-7344 across all systems
  - EFI System Partition integrity verification and monitoring implementation

RETURN >>>

- Bootloader verification and backup procedures for rapid restoration capabilities
- UEFI-aware backup solutions deployment for firmware-level recovery
- Enhanced incident response procedures for firmware-level compromise scenarios

Tactical Intelligence

- **Mitigation Strategies**
  - Apply UEFI firmware updates addressing CVE-2024-7344 with emergency priority
  - Enable UEFI Secure Boot verification across all endpoint and server systems
  - Implement EFI System Partition monitoring for unauthorized file creation
  - Deploy measured boot with TPM verification capabilities
  - Establish UEFI firmware update management processes with security verification
- **Preventive Measures**
  - Bootloader integrity checking mechanisms with cryptographic verification
  - EFI System Partition access monitoring and unauthorized modification detection
  - UEFI firmware modification tracking and alerting
  - TPM-based attestation for boot process integrity verification
  - Regular EFI System Partition backup and integrity baseline establishment
- **Detection Commands**
  - bcdedit /enum all (enumerate boot configuration and detect modifications)
  - dir /s C:\EFI\ (scan EFI System Partition for unauthorized files)
  - mountvol (identify EFI System Partition mount points for monitoring)
  - Get-SecureBootUEFI (PowerShell verification of Secure Boot status)

Threat Hunting Hypotheses

UEFI Secure Boot Bypass Detection

Hypothesis: Attackers exploit CVE-2024-7344 vulnerability to load malicious EFI applications through signed but vulnerable Microsoft applications, bypassing Secure Boot protections.

Investigation Steps

1. 1. Monitor EFI System Partition for unauthorized "cloak.dat" file creation
2. 2. Detect "reloader.efi" execution outside normal firmware update processes
3. 3. Identify modifications to EFI boot entries pointing to non-standard applications
4. 4. Track UEFI Secure Boot status changes or disable attempts
5. 5. Analyze firmware update logs for unauthorized or suspicious modification attempts

Firmware-Level Persistence Establishment

Hypothesis: Ransomware creates comprehensive tracking mechanisms and backup files on EFI System Partition to maintain persistence across operating system reinstallation attempts.

Investigation Steps

1. Scan EFI System Partition for unauthorized tracking files and backup bootloaders
2. Monitor legitimate bootloader replacement with malicious variants
3. Detect victim-specific installation key generation artifacts
4. Identify unusual EFI application installations outside vendor update channels
5. Track EFI System Partition file creation and modification patterns for anomalies

RETURN >>>

Tactical Guidance

MFT Encryption and System Compromise

Hypothesis: Attackers target NTFS Master File Table structures for encryption while displaying fake system maintenance messages to delay detection and response.

Investigation Steps

1. Monitor for fake CHKDSK message display during non-scheduled maintenance windows
2. Detect Salsa20 encryption operations targeting filesystem metadata structures
3. Identify unauthorized Bitcoin payment demands or ransom note creation
4. Track system boot failures or corruption patterns consistent with MFT encryption
5. Analyze filesystem integrity for Master File Table modification or corruption indicators

Sources

- **ESET:** ESET Research Discovers HybridPetya Ransomware Secure Boot Bypass
- **WeLiveSecurity:** Introducing HybridPetya - Petya/NotPetya Copycat with UEFI Secure Boot Bypass
- **WeLiveSecurity:** HybridPetya - Petya/NotPetya Copycat with a Twist
- **Infosecurity Magazine:** HybridPetya Mimics NotPetya with UEFI Bypass
- **Bank Info Security:** HybridPetya Crypto-Locker Outsmarts UEFI Secure Boot
- **Petri:** HybridPetya Ransomware Analysis
- **SC World:** UEFI Secure Boot Circumvented by Novel HybridPetya Ransomware
- **The Hacker News:** New HybridPetya Ransomware Bypasses UEFI Secure Boot

AI MODEL SUPPLY CHAIN ATTACKS - NAMESPACE REUSE

Overview & Impact

Model namespace reuse attacks target AI model supply chains through systematic exploitation of deleted or transferred namespaces on platforms like Hugging Face, enabling attackers to register malicious models with identical names to legitimate models. Security researchers successfully demonstrated this attack against Google Vertex AI Model Garden and Microsoft Azure AI Foundry, achieving infrastructure access and Azure endpoint permissions through embedded reverse shell code. The attack affects thousands of open-source repositories containing vulnerable model references and creates extensive transitive supply chain risks across software ecosystems.

The attack exploits fundamental architectural weaknesses in model referencing systems that automatically trust models based on "Author/ModelName" format identifiers without verifying ownership continuity after account changes. Automated scanning tools identify widespread vulnerable model dependencies requiring systematic remediation across software development environments. Organizations implementing AI capabilities face immediate risk exposure, particularly those deploying automated model deployment pipelines that lack version verification controls.

RETURN >>>

Unauthorized model deployment enabling arbitrary code execution and infrastructure access

- Compromise of Google Vertex AI and Microsoft Azure AI platforms through namespace exploitation
- Thousands of open-source repositories containing vulnerable model references
- Transitive supply chain risks affecting organizations without direct AI development focus
- Potential for systematic AI supply chain targeting with minimal operational overhead

Threat Implications

- AI model supply chain attacks represent fundamental shift in software dependency threat landscape
- Major cloud providers lack sufficient model authenticity verification controls
- AI dependency security requires equivalent rigor to traditional software supply chain controls
- Transitive supply chain risks extend beyond direct AI development teams

Notable Campaigns

- Google Vertex AI Model Garden successful compromise demonstration
- Microsoft Azure AI Foundry endpoint permission acquisition
- Thousands of vulnerable model references identified across open-source repositories
- Hugging Face platform namespace exploitation for model replacement

MITRE ATT&CK Framework Mapping

- **T1195.002 - Supply Chain Compromise:** Software Supply Chain
- **T1078.004 - Valid Accounts:** Cloud Accounts
- **T1059.006 - Command and Scripting Interpreter:** Python
- **T1543.003 - Create or Modify System Process:** Windows Service
- **T1105 - Ingress Tool Transfer**

Observations

- Model namespace deletion provides reliable attack vector for malicious model registration
- Automated deployment pipelines lack model authenticity verification across major cloud platforms
- Model references embedded in unexpected locations including comments and configuration files
- Reverse shell code successfully executes within model deployment environments
- Thousands of repositories contain vulnerable model dependencies requiring systematic remediation
- AI platform security controls insufficient for model supply chain threat detection
- Automated scanning enables systematic identification of vulnerable model references

RETURN >>>

Guidance

Strategic Intelligence

- **Trend**
  - AI model supply chain attacks emerging as critical new vector for infrastructure compromise
  - Cloud providers investing insufficient resources in model authenticity verification controls
  - AI dependency proliferation across software projects creating widespread exposure
  - Threat actors developing automated techniques for systematic AI supply chain targeting
- **Business Risk Context**
  - Organizations implementing AI face immediate supply chain compromise risks
  - AI model dependencies require equivalent security rigor to traditional software libraries
  - Transitive AI supply chain risks affect organizations beyond direct AI development
  - AI platform compromise enables persistent infrastructure access and data exfiltration

Operational Intelligence

- **Threat Vectors**
  - Deleted model namespace registration for malicious model deployment
  - Automated deployment pipeline exploitation through model reference attacks
  - Reverse shell code embedding within model files for infrastructure access
  - Cloud AI platform permission escalation through malicious model deployment
  - Transitive dependency exploitation through vulnerable model references
- **Monitoring & Detection Gaps**
  - AI model dependency resolution processes lack comprehensive security monitoring
  - Model authenticity verification insufficient across cloud AI platforms
  - Vulnerable model references in code repositories and configuration undetected
  - Model deployment operations not monitored for unauthorized access or execution
  - AI platform permission changes and endpoint access not tracked for anomalies
- **Response Actions**
  - Immediate AI model inventory and dependency auditing across development and production environments
  - Model verification processes implementation before deployment operations
  - Cloud AI platform configuration hardening with authenticity checking requirements
  - Internal model repository mirroring for trusted model access
  - Enhanced monitoring for model deployment and AI platform access activities

Tactical Intelligence

- **Mitigation Strategies**
  - Pin all models to specific commit hashes rather than latest versions
  - Clone trusted models to internal repository infrastructure
  - Implement model verification processes before deployment operations
  - Configure Google Vertex AI with model verification requirements
  - Enable Microsoft Azure AI Foundry model authenticity checking

RETURN >>>

- **Preventive Measures**
  - Scan software repositories for AI model references in code and configuration
  - Deploy model scanning tools within CI/CD pipeline automation
  - Monitor cloud AI platform model deployments for unauthorized changes
  - Implement immutable model references with cryptographic verification
  - Establish AI dependency management processes equivalent to software supply chain controls
- **Detection Commands**
  - grep -r "huggingface.co" . --include="*.py" --include="*.js" --include="*.json" (identify model references)
  - find . -name "*.py" -exec grep -l "from_pretrained" {} \; (locate model loading operations)
  - grep -r "models/" . | grep -E "Author/ModelName" (detect vulnerable reference patterns)
  - git log --grep="model" --oneline (track model-related changes)

Threat Hunting Hypotheses

Model Namespace Hijacking

Hypothesis: Attackers systematically monitor for deleted model namespaces and register malicious models with identical names to target automated deployment pipelines.

Investigation Steps

1. Monitor model repository platforms for namespace deletion and immediate re-registration patterns
2. Detect model deployment attempts from recently registered accounts or suspicious namespace changes
3. Identify hardcoded model references without version pinning in application code and configuration files
4. Track model download operations for unexpected source changes or authenticity verification failures
5. Analyze model repository activity logs for systematic namespace monitoring and registration activities

AI Platform Infrastructure Compromise

Hypothesis: Malicious models deployed through compromised namespaces establish persistent infrastructure access and enable lateral movement within cloud AI platforms.

Investigation Steps

1. Monitor cloud AI platform deployments for unauthorized model installations and permission acquisitions
2. Detect reverse shell or command execution activities originating from model deployment processesdentify unexpected network connections or data exfiltration from AI model hosting infrastructure
3. Track model-triggered infrastructure access logs for anomalous authentication or authorization patterns
4. Analyze AI platform endpoint permission changes following model deployment activities

Transitive AI Dependency Exploitation

Hypothesis: Organizations unknowingly deploy compromised AI models through transitive dependencies in software projects that incorporate AI capabilities without direct AI development focus.

Investigation Steps

1. Scan software repositories for AI model references in comments, documentation, and configuration files
2. Detect automatic model fetching operations during application deployment or runtime initialization
3. Identify AI model dependencies in CI/CD pipelines and deployment automation scripts
4. Monitor application behavior for unexpected model loading or AI inference operations
5. Track software dependency trees for AI model references and potential vulnerability exposure

RETURN >>>

Tactical Guidance

Sources

- **Unit 42:** Model Namespace Reuse Attack Analysis
- **NIST:** Characterization of AI Model Configurations for Model Reuse
- **Unit 42:** Model Namespace Reuse Technical Details
- **ACM Digital Library:** AI Model Reuse Research
- **TraxTech:** Model Namespace Reuse Attacks Compromise Google and Microsoft AI Platforms
- **CodeKeeper:** Model Namespace Reuse Attack Exploits
- **Security Week:** AI Supply Chain Attack Method Demonstrated Against Google Microsoft Products
- **HackRead:** Model Namespace Reuse Flaw in AI Models - Google Microsoft
- **SC World:** Hugging Face Model Namespace Reuse Poses AI Supply Chain Risk

RETURN >>>

Contact the Pellera Threat Intel Group at getsecure@pellera.com

pellera.com

A PELLERA PODCAST

Edge of I.T.