CloudCommand

Software-installation

CloudCommand GmbH chr.schumacher@gmx.tm

Grundlegende Bedeutung

Warum ist das Thema so wichtig?

- Softwareinstallation = potentielles Einfallstor für Angriffe
- Durchdachte Installation verhindert Malware, Datenlecks und Systemausfälle

„Besser gründlich prüfen, als hinterher doppelt flicken!“

Risikofaktoren

Beispiele:

- **Unbekannte Quellen** (Raubkopien, inoffizielle Mirror-Sites)
- **Schadsoftware in Installationspaketen** (Trojaner, Spyware)
- **Falsche Berechtigungen** (Installations-Setup verlangt Admin-Rechte ohne Notwendigkeit)

Compliance & rechtliche Vorgaben

- **Stichworte:** DSGVO, ISO 27001, BSI-Grundschutz
- **Einhaltung von Vorgaben:**
  - Richtige Handhabung von Lizenzvereinbarungen
  - Protokollierung, wer wann was installiert

Vertrauenswürdige Quellen und digitale Signaturen

Checkliste:

1. Offizielle Hersteller-Websites oder Repositories verwenden
2. Prüfen von Code Signing (z. B. Signaturen bei ausführbaren Dateien)
3. Sicherheitswarnungen oder Zertifikats-Fehler ernst nehmen

Hash-Prüfung und Integrität der Software

Theorie:

- Hashes (z. B. MD5, SHA-256) dienen dem Abgleich von Datei-Integrität

Praxis:

- Heruntergeladenes Installationspaket mit Hersteller-Hash vergleichen
- Tools wie sha256sum (Linux) oder Get-FileHash (Windows PowerShell) nutzen

Berechtigungen und Least Privilege

- **Idee:** Nur so viele Rechte wie nötig vergeben.
- **Warum?**
  - Reduziert mögliche Schäden, sollte sich Malware einschleichen.
  - Minimiert Fehlkonfigurationen, wenn nicht jeder volle Admin-Rechte hat.

Tipp: *Eine Installation als normaler Benutzer statt Administrator ist meist sicherer.*

Sandbox- und Testumgebungen

Vorteile:

- Installation gefahrlos testen
- Fehlverhalten beobachten, bevor es ins Livesystem geht

Tools:

- Virtuelle Maschinen (VirtualBox, VMware)
- Windows Sandbox

Supply-Chain-Angriffe

- **Erklärung:** Angriffe über kompromittierte Software-Lieferketten (z. B. SolarWinds-Fall)
- **Schutzmaßnahmen:**
  - Vertrauen in Drittanbieter sorgfältig prüfen
  - Regelmäßige Updates und Überwachung von Code-Repositories
  - Zero-Trust-Ansatz in Entwicklungs- und Beschaffungsprozessen

Hinweis: *Der Feind sitzt nicht immer draußen, manchmal nistet er sich unbemerkt in Updates ein!*

Patches und Updates

- **Gute Praxis:**
  - Regelmäßige Patch-Management-Zyklen (Patch-Tuesday etc.)
  - Schnelle Installation kritischer Sicherheitsupdates
- **Risiko:** Nicht gepatchte Software = offene Scheunentor für Angreifer.

Secure Configuration

- Werkseinstellungen sind oft unsicher.
- **Lösung:**
  - Passwörter und Standard-Ports ändern
  - Unnötige Features oder Dienste deaktivieren
  - Sicherheitskonfiguration an Unternehmensstandards anpassen

Bittere Realität: *Schätzungsweise mehr als 50% der IoT-Geräte laufen mit Admin/Admin o. Ä.!*

Containerisierung und Sicherheit

- **Chancen:**
  - Isolierung einzelner Dienste
  - Schnelle Deployments
- **Risiken:**
  - Unsichere Images, falsche Konfiguration (privilegierte Container)
  - Fehlende Monitoring- und Patch-Strategien

Tipp: *Zieht euch offizielle Images oder erstellt eigene, geprüfte Basis-Images.*

Rollback-Strategien & Backups

- **Warum Rollbacks?**
  - Fehlgeschlagene Installation kann System destabilisieren
  - Schadsoftware oder inkompatible Updates erfordern schnelles Zurückrollen
- **Best Practices:**
  - Vor dem Update/Installation ein Backup machen
  - Dokumentierte Rollback-Anleitung

DevSecOps-Prinzipien beim Installieren

- **Definition:** Security als integraler Bestandteil des gesamten Entwicklungs- und Deployment-Prozesses.
- **Wichtige Punkte:**
  - Frühe Sicherheits-Reviews in der Pipeline
  - Automatisierte Code- und Vulnerability-Scans
  - Zusammenarbeit von Entwicklern, Ops und Security

Security-Tools & Hilfsmittel

- **Beispiele:**
  - Antivirus/EDR-Lösungen (z. B. Windows Defender, CrowdStrike)
  - Honeypots zum Erkennen von Angriffsversuchen
  - Patchmanagement-Tools (z. B. WSUS, Ansible)

Häufige Fehler und wie man sie vermeidet

- **Top 5 Fehler:**
  - Installationsdateien aus dubiosen Quellen
  - „Next, Next, Finish“ ohne Lesen der Meldungen
  - Unvollständiges Patchen (alte Versionen verbleiben)
  - Fehlende Dokumentation des Installationsvorgangs
  - Keine Ausweich- oder Testumgebung

Fallbeispiel 1: SolarWinds Orion (2020)

- **Überblick:**
  - Gilt als einer der bekanntesten Supply-Chain-Angriffe der letzten Jahre.
  - Orion-Software von SolarWinds, die von Tausenden Kunden weltweit genutzt wurde (darunter Regierungsbehörden, Konzerne).

Kernfaktor: *Angreifer kompromittierten das Update-System von SolarWinds, um eine manipulierte Softwareversion an Endkunden auszuliefern.*

Fallbeispiel 1: SolarWinds Orion (2020)

- **Wie kam es zur Kompromittierung?**
  - Angreifer verschafften sich Zugriff auf die Build-Umgebung (Software-Entwicklungsprozess).
  - Manipulierte Updates wurden signiert und sahen daher ‚offiziell' aus.
  - Kunden installierten das Update und importierten damit direkt Backdoors in ihre Systeme.

Fallbeispiel 1: SolarWinds Orion (2020)

- **Betroffene:** US-Behörden, internationale Konzerne, Sicherheitsunternehmen
- **Folgen:**
  - Zugriff der Angreifer auf interne Netzwerke, E-Mails, vertrauliche Daten
  - Hohe Kosten für Incident Response, Forensik und Systembereinigungen
  - Vertrauensverlust in die gesamten Software-Lieferketten

Fallbeispiel 1: SolarWinds Orion (2020)

- **Schlüsselerkenntnisse:**
  - Strenge Kontrolle der Build-Prozesse (Code-Integrität, Zugriffskontrollen)
  - Vertrauen ist gut, aber ständige Überprüfung (z. B. automatisierte Security-Scans, Code-Reviews) ist besser
  - Schnelle Reaktion und umfassendes Monitoring, um verdächtige Updates frühzeitig zu entdecken

Fallbeispiel 2: CCleaner-Hack (2017)

- **Überblick:**
  - CCleaner – ein beliebtes Wartungstool für Windows, millionenfach heruntergeladen.
  - 2017 wurde eine offiziell signierte Version als Trojaner verteilt.

Kernfaktor: *Die gehackte Version gelang in den offiziellen Download-Kanal, sodass Nutzer beim regulären Update die Malware installierten.*

Fallbeispiel 2: CCleaner-Hack (2017)

- **Was passierte genau?**
  - Angreifer infizierten den Build-Server bei Piriform/Avast (CCleaner-Herausgeber).
  - Das trojanisierte Installationspaket wurde auf die offizielle Downloadseite gestellt.
  - Nutzer luden mit bestem Gewissen „das neueste Update“ – und erhielten Schadcode.

Fallbeispiel 2: CCleaner-Hack (2017)

- **Lehren für Softwareinstallation:**
  - Vertrauenswürdige Marken sind nicht automatisch sicher – Angreifer zielen oft gerade auf populäre Software.
  - Digitales Signieren allein reicht nicht, wenn der Build-Prozess selbst kompromittiert wird.
  - Integrität laufend überwachen, z. B. mithilfe zusätzlicher Hash-/Signaturchecks oder Abweichungsanalysen.

Informationsquellen:

NIST – National Institute of Standards and Technology

- Special Publication 800-53 – Security and Privacy Controls for Information Systems and Organizations

BSI – Bundesamt für Sicherheit in der Informationstechnik

- BSI-Standards und IT-Grundschutz-Kataloge (bsi.bund.de)

OWASP – Open Web Application Security Project

- OWASP Secure Software Practices (owasp.org)

SolarWinds- / SUNBURST-Angriff

- CISA (2021): Analysis Reports zum SolarWinds-Orion Vorfall (cisa.gov/supply-chain-compromise)
- SolarWinds Blog: „An Update on the Ongoing Investigation" (2020)

Informationsquellen:

CCleaner-Hack (2017)

- Talos Intelligence: „CCleaner Command and Control“ (blog.talosintelligence.com)
- Offizielle Avast/Piriform Stellungnahmen

Microsoft Docs – Windows Security Guidelines (docs.microsoft.com)

Rapid7 & Kaspersky – Berichte zu IoT-Sicherheit und Supply-Chain-Angriffen

Diverse Branchenblogs / Fachartikel

- Heise Security, Golem, Ars Technica

CloudCommand