Sichere automatisierte Ver-/Entschlüsselung für DevSecOps

Stefan Schumacher

Chemnitzer Linux-Tage 2024

$Id: CLT2024-SichereDevSecOpsVerschluesselung.tex,v 1.2 2024/03/17 15:32:09 stefan Exp $

Inhaltsverzeichnis

1 Hintergrund

2 Secret Sharing und Schlüsselaustausch

3 Nichtabstreitbarkeit

4 Literatur

Hintergrund

- Dataport Cloud Security Architect, Schwerpunkt Kryptographie
- https://www.dphoenixsuite.de/
- Open Source basierter Online-Arbeitsplatz
- U.a. mit Dateiablage, Video-Konferenzen, Instant Messaging, Online-Office
- Z.B. für Schulen, Gesundheitswesen, Robert-Koch-Institut, weitere Bedarfsträger
- Datenschutz, BSI Grundschutz Hoch, Evaluation Verschlusssache VS NfD

Hintergrund

- Schulen gehen montags 07:00 Uhr online und Freitags 14:30 offline
- Skalierbarkeit notwendig und durch Containerisierung möglich
- Agile, schnelle Entwicklung und Bereitstellung $\rightsquigarrow$ DevOps

DevOps

- Prozesse (und Kultur) der Softwareentwicklung (Dev) und SysAdmin (Ops)
- Continuous Integration (CI) und Continuous Delivery (CD) (Gitlab, Argo CD)
- Infrastructure as a Code (Terraform, Ansible)
- Software-Configuration-Management (Ansible)
- Versionsverwaltung (Gitlab)
- Schnelle Release-Zyklen, Verschränkung und Zusammenarbeit von Dev und Ops

DevSecOps

- Veraltete Sicherheitspraktiken scheitern an DevOps
- DevSecOps soll Sicherheit von Anfang an mitdenken
- Integrierte Sicherheit im Zyklus der Softwareentwicklung
- Risikotoleranz bestimmen $\rightsquigarrow$ Kosten-Nutzen-Analyse
- Sicherheitsautomatisierung wo möglich durch entsprechende Tools

DevSecOps

Grundideen

- Zero-Trust-Architektur
- Microservices
- Container (Docker, Kubernetes) insbesondere zur Skalierbarkeit
- Versionierung und Automatisierung durch Gitlab/Github
- Automatisierung und Sicherheit in der CI/CD-Pipeline
- Authentifikation und Autorisation
- Transportverschlüsselung und Zertifikatsmanagement

Grundfragen

VS NfD, Cloud nicht im eigenen Rechenzentrum:

1. Wie manage ich sensible Daten außerhalb des Clusters, also bevor sie in der Build- und Deploy-Phase ausgerollt werden?
2. Wie sichere ich meine sensiblen Daten im Cluster, wenn auf sie durch Applikationen zugegriffen wird?
3. Wie stelle ich ggf. Key Escrow sicher?
4. Wie kann ich revisionssicher protokollieren, wer wann (wieso, weshalb, warum) auf sensible Daten zugegriffen hat?
5. Es gibt noch keine Software, die diese Anforderung alle erfüllt
6. Zwischenstand der Evaluation VS NfD

Grundfragen

VS NfD, Cloud nicht im eigenen Rechenzentrum:

1. Wie manage ich sensible Daten außerhalb des Clusters, also bevor sie in der Build- und Deploy-Phase ausgerollt werden?
2. Wie sichere ich meine sensiblen Daten im Cluster, wenn auf sie durch Applikationen zugegriffen wird?
3. Wie stelle ich ggf. Key Escrow sicher?
4. Wie kann ich revisionssicher protokollieren, wer wann (wieso, weshalb, warum) auf sensible Daten zugegriffen hat?
5. Es gibt noch keine Software, die diese Anforderung alle erfüllt
6. Zwischenstand der Evaluation VS NfD

Schutzziele und Zustände

- Vertraulichkeit
- Integrität
- Authentizität
- Nichtabstreitbarkeit

Kryptographie

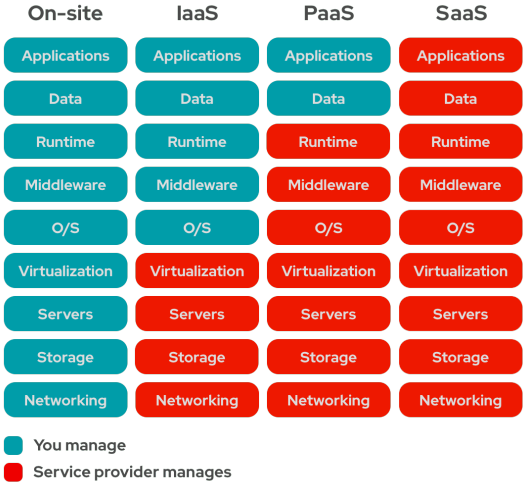
1. Data at Rest
2. Data in Transfer
3. Data in Use

Kubernetes Secrets

Ist-Zustand

- Kubernetes Secrets: Base64 codiert
- Nicht verschlüsselt!
- Secrets müssen verschlüsselt werden bevor sie in den Cluster kommen
  z.B. mit KSOPS, Mozilla SOPS, Bitnami Sealed Secrets
- Secrets müssen im Cluster vor Gebrauch entschlüsselt werden
- Schlüsselmanagement?
- »Key material accessible from control plane host.«
- Schlüssel liegen im Dateisystem des API Server Hosts
  `/etc/kubernetes/static-pod-resources/kube-apiserver-pod-$VERSION/secrets/encryption-config/encryption-config`

Die Cloud ist **immer** anderer Leute Computer



Schlüssel liegen im Dateisystem des API Server Hosts

Kubernetes Secrets

- Schlüssel liegen im Dateisystem des API Server Hosts, das ist eher suboptimal
- Festplattenverschlüsselung aktivieren
- Wer gibt das Passwort ein?
- Turnschuhadmin ist schnell und agil ...
- Keys von einem Server holen
- TLS ist sicher!

Kubernetes Secrets

- Schlüssel liegen im Dateisystem des API Server Hosts, das ist eher suboptimal
- Festplattenverschlüsselung aktivieren
- Wer gibt das Passwort ein?
- Turnschuhadmin ist schnell und agil ...
- Keys von einem Server holen
- TLS ist sicher!
- TLS ist sicher? Heartbleed?
- Wie bekomme ich Schlüsselmaterial sicher durchs Netz?
- Gar nicht.

Kubernetes Secrets

- Schlüssel liegen im Dateisystem des API Server Hosts, das ist eher suboptimal
- Festplattenverschlüsselung aktivieren
- Wer gibt das Passwort ein?
- Turnschuhadmin ist schnell und agil ...
- Keys von einem Server holen
- TLS ist sicher!
- TLS ist sicher? Heartbleed?
- Wie bekomme ich Schlüsselmaterial sicher durchs Netz?
- Gar nicht.
- Ich bekomme also kein Schlüsselmaterial sicher durchs Netz.
- Was wenn wir keine Schlüssel übertragen, sondern Schlüssel ableiten?

Kubernetes Secrets

- Schlüssel liegen im Dateisystem des API Server Hosts, das ist eher suboptimal
- Festplattenverschlüsselung aktivieren
- Wer gibt das Passwort ein?
- Turnschuhadmin ist schnell und agil ...
- Keys von einem Server holen
- TLS ist sicher!
- TLS ist sicher? Heartbleed?
- Wie bekomme ich Schlüsselmaterial sicher durchs Netz?
- Gar nicht.
- Ich bekomme also kein Schlüsselmaterial sicher durchs Netz.
- Was wenn wir keine Schlüssel übertragen, sondern Schlüssel ableiten?

Inhaltsverzeichnis

1 Hintergrund

2 Secret Sharing und Schlüsselaustausch

3 Nichtabstreitbarkeit

4 Literatur

Diffie-Hellman-KEX

- Vereinbarung von Schlüsselpaaren aus öffentlichen und privaten Schlüsseln über unsichere Kanäle
- Alice und Bob tauschen öffentliche Schlüssel aus
- über unsicheren Kanal möglich
- Secret kann mittels asymmetrischer Kryptographie übertragen werden
- Secret liegt aber immer noch auf dem Server
- Key Escrow problematisch

geteilte Geheimnisse

- Idee: ich verteile ein Geheimnis auf mehrere Geheimnisträger
- Noch nie hat ein X irgendwo, irgendwann einen bedeutenden Punkt markiert: Schatzkarte zerteilen

```
echo Passwort | fold -2
Pa
ss
wo
rt

echo Passwort | sha512sum | awk '{print $1}' |  fold -w 32
d0ec31286c4370b962c2fffa2d0b442d
fa7a9346347edfbb2c52a5a0908d4253
9d59d14069794882fa12101ec7183ce2
57bc791fa9023b731af9e93544f6f513
```

geteilte Geheimnisse

Schwellwert-Kryptosystem

- 4 Teil-Geheimnisse, alle 4 zusammen ergeben das Original-Geheimnis
- 1/2/3 Geheimnisträger allein nicht ausreichend
- Was wenn Alice, Charlie und David zusammenarbeiten?
- Was wenn Alice ausfällt?
- Idee: m aus n Geheimnisträgern sollen ausreichen
- Reduktion des Schlüsselraumes auf den Schwellwert, z.B. 3 aus 4 Geheimnissen
- Einfaches Beispiel: Schnittpunkt dreier Ebenen im $\mathbb{R}^3$

Shamir's Secret Sharing

- Stützstellen eines Polynoms repräsentieren Shares
- $f(x) = s + a_1 \cdot x + a_2 \cdot x^2 + \cdots + a_{t-1} \cdot x^{t-1} mod\ p$
- s ist das Geheimnis
- n Wertepaare der Form $x_i, s_i = f(x_i))$ mit $x_i \neq 0$
- Fundamentalsatz der Algebra:
  t Wertepaare werden zur Rekonstruktion des Polynoms benötigt
- $t - 1$ Wertepaare können kompromittiert werden bzw
- Effiziente Rekonstruktion des Polynoms mittels Lagrange-Interpolation

McCallum-Relyea-Schlüsselaustausch I

- Schlüsselaustauschverfahren, bringt Diffie-Hellman KEX und Shamir's Secret Sharing zusammen
- Daten werden mit einem Data/Transport Encryption Key (DEK bzw. TEK) verschlüsselt (Secret)
- DEK bzw. TEK werden mit einem weiteren sog. Key Encryption Key (KEK) verschlüsselt
- KEK kann im McCallum-Relyea-Schlüsselaustausch *regeneriert* werden (ephemeraler Schlüssel)
- ohne den KEK tatsächlich zu übertragen
- Datenübertragung über ungesicherten Kanal möglich
- Key Escrow auch möglich

McCallum-Relyea-Schlüsselaustausch I

1. Der **Server** generiert ein langlebiges Elliptic-Curve-Schlüsselpaar mit dem privaten Schlüssel s und dem öffentlichen Schlüssel $S = [s]G$.
2. Der **Client**, der eine Nachricht M schützen möchte, generiert seinen eigenen privaten Schlüssel c und den öffentlichen Schlüssel $C = [c]G$.
3. Der Client fordert über einen unverschlüsselten Kanal einen Schlüssel vom Server an.
4. Der Server antwortet mit S, signiert mit s.
5. Der Client validiert den Server-Schlüssel und speichert den Hash des Server-Schlüssels.
6. Der Client führt einen Teil des ECDH-Austauschs durch, um $K = [c]S = [cs]G$ zu berechnen. K wird verwendet, um M über einen symmetrischen Algorithmus zu verschlüsseln.
7. Der Client verwirft K und c und behält nur C.
8. Wenn der Client M entschlüsseln möchte, erstellt er ein ephemeres Schlüsselpaar e und berechnet $X = C + E$.

McCallum-Relyea-Schlüsselaustausch II

9. Der Client sendet X an den Server.
10. Der Server berechnet $Y = [s]X$ und sendet es zurück an den Client.
11. Der Client berechnet $Z = [e]S$ und erhält so den ursprünglichen Schlüssel K zurück.
12. Mit K kann der Client M entschlüsseln.
13. $S \in \mathbb{R}[1, p-1]$, $s = g^S \rightarrow s$
14. $C \in \mathbb{R}[1, p-1]$, $c = g^C = g^{SC} = sC$
15. Verwerfen: K, C, Behalten: s, c
16. **Schlüsselaustausch**: $E \in \mathbb{R}[1, p-1]$, $e = g^E$, $x = c + e$, $x \rightarrow y = x^S$, $y = y - sE$
17. Weil: $K = g^{CS} + g^{ES} - g^{SE}$
18. Um c privat zu halten, müssen e und E privat sein.[1]

[1] Der Beweis ist trivial und bleibt dem Leser zur Übung überlassen.

Anwendung

LUKS FDE mit Clevis/Tang

- McCallum-Relyea-Schlüsselaustausch: LUKS FDE, Clevis, Tang
- Clevis vollzieht per Netzwerk mit Tang-Server McC-R-KEX
- Tang-Server muss erreichbar sein, z.B. im Firmennetz
- NBDE: Network Bound Disk Encryption
- Policy Based (Disk) Encryption möglich
- Verkettung von Authentifikationsmechanismen möglich (TPM2 && Yubikey && Clevis)

Inhaltsverzeichnis

1 Hintergrund

2 Secret Sharing und Schlüsselaustausch

3 Nichtabstreitbarkeit

4 Literatur

Nichtabstreitbarkeit

- Urheberschaft feststellen
- Wer hat wann von wo welche Aktion ausgelöst?
- Worst Case: Innentäter mit `root`-Rechten
- Revisionssicher?
- `/dev/tty` nach `/dev/printer` umleiten
- Bessere Optionen?
- Ich bin `root`, ich darf das!

Signaturen

- Logdatei/-Eintrag mit kryptographischer Signatur sichern
- `gpg -a -output log0.sig -detach-sign log0.txt`
- Schlüsselmanagement $\rightsquigarrow$ Komplexität
- Einfachere Lösung: nur einen Hash bilden
- `sha256sum log0`
- Kann aber wieder manipuliert werden

Verkettete Signaturen

Merkle-Trees

- Hashes hashen
- In einer effizienten Datenstruktur $\rightsquigarrow$ Baum
- Merkle-Tree: Binärbaum, in dem jeder Knoten den Hash über alle seine Unterbäume bildet
- Jede Änderung an jeder Stelle pflanzt sich bis zum Wurzel-Hash fort
- Fällt damit schnell (und effizient) auf

Verteilte Signatur-Ketten

- Verteilte Datenbank mit beliebig vielen Knoten (bzw. Rechnern)
- Merkle-Tree wird auf jedem Knoten synchronisiert
- Jede schreibende Transaktion ändert die Hashes bis zum Wurzel-Hash
- Jede Änderung fällt sofort auf
- Und kann mit allen anderen Knoten verglichen werden
- Eine Art verteiltes Protokoll
- Manipulationssicher und Effizient

Fragen?

- stefan.schumacher2 <at> dataport.de

Inhaltsverzeichnis

1 Hintergrund

2 Secret Sharing und Schlüsselaustausch

3 Nichtabstreitbarkeit

4 Literatur

Brickell, E. F. (1989). Some ideal secret sharing schemes. In *Workshop on the Theory and Application of of Cryptographic Techniques* (S. 468–475). Springer.

Stadler, M. (1996). Publicly verifiable secret sharing. In *International Conference on the Theory and Applications of Cryptographic Techniques* (S. 190–199). Springer.

Shamir, A. (1979). How to share a secret. *Communications of the ACM*, *22*(11), 612–613.

Shamir, A. (1985). Identity-based cryptosystems and signature schemes. In *Advances in Cryptology: Proceedings of CRYPTO 84 4* (S. 47–53). Springer.

Fiat, A. & Shamir, A. (1986). How to prove yourself: Practical solutions to identification and signature problems. In *Conference on the theory and application of cryptographic techniques* (S. 186–194). Springer.

Canetti, R., Chen, Y., Holmgren, J., Lombardi, A., Rothblum, G. N., Rothblum, R. D. & Wichs, D. (2019). Fiat-Shamir: from practice to theory. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing* (S. 1082–1090).

Voggenreiter, M., Angermeir, F., Moyón, F., Schöpp, U. & Bonvin, P. (2024). Automated Security Findings Management: A Case Study in Industrial DevOps. *arXiv preprint arXiv:2401.06602.*

Moyón, F., Angermeir, F. & Mendez, D. (2024). Industrial Challenges in Secure Continuous Development. *arXiv preprint arXiv:2401.06529.*

Lykousas, N. & Patsakis, C. (2023). Tales from the Git: Automating the detection of secrets on code and assessing developers' passwords choices. arXiv: 2307.00892 [cs.SE]

Pecka, N., ben Othmane, L. & Valani, A. (2022). Making Secure Software Insecure without Changing Its Code: The Possibilities and Impacts of Attacks on the DevOps Pipeline. arXiv: 2201.12879 [cs.CR]

Pecka, N., ben Othmane, L. & Valani, A. (2022). Making Secure Software Insecure without Changing Its Code: The Possibilities and Impacts of Attacks on the DevOps Pipeline. arXiv: 2201.12879 [cs.CR]

Shamim, M. S. I., Bhuiyan, F. A. & Rahman, A. (2020). XI Commandments of Kubernetes Security: A Systematization of Knowledge Related to Kubernetes Security Practices. *CoRR, abs/2006.15275*. arXiv: 2006.15275. Verfügbar unter https://arxiv.org/abs/2006.15275

McCullagh, A. & Caelli, W. (2000). Non-repudiation in the digital environment.

Fang, W., Chen, W., Zhang, W., Pei, J., Gao, W. & Wang, G. (2020). Digital signature scheme for information non-repudiation in blockchain: a state of the art review. *EURASIP Journal on Wireless Communications and Networking, 2020*(1), 1–15.

Roe, M. (2010). *Cryptography and evidence*. University of Cambridge, Computer Laboratory.

Schlatt, V., Schweizer, A., Urbach, N. & Fridgen, G. (2016). *Blockchain: Grundlagen, Anwendungen und Potenziale*. Fraunhofer-Institut für Angewandte Informationstechnik. Verfügbar unter https://www.fit.fraunhofer.de/content/dam/fit/de/documents/Blockchain_WhitePaper_Grundlagen-Anwendungen-Potentiale.pdf

Kaucher, A. (2018). *Blockchain: Funktionsweise und Applikationsmöglichkeiten*. Hochschule Bonn-Rhein-Sieg. Verfügbar unter https://www.h-brs.de/files/2018-01-14_seminar_alexander_kaucher_final.pdf

El Ioini, N. & Pahl, C. (2018). A review of distributed ledger technologies. In *On the Move to Meaningful Internet Systems. OTM 2018 Conferences: Confederated International Conferences: CoopIS, C&TC, and ODBASE 2018, Valletta, Malta, October 22-26, 2018, Proceedings, Part II* (S. 277–288). Springer.

Voggenreiter, M., Angermeir, F., Moyón, F., Schöpp, U. & Bonvin, P. (2024). Automated Security Findings Management: A Case Study in Industrial DevOps. *arXiv preprint arXiv:2401.06602*.

https://www.snia.org/sites/default/files/SDC15_presentations/security/NathanielMcCallum_Network_Bound_Encryption.pdf
https://people.redhat.com/pladd/NYRHUG_NBDE_2020-02.pdf