Engineer

Privileg

🎟 Vom System an eine Person oder einen Prozess gewährtes Recht

Was es ist

Ein Privileg ist die Erlaubnis, etwas zu tun, das ein System sonst verbietet. Eine Person oder ein laufendes Programm besitzt sie, weil eine befugte Stelle sie erteilt hat: fremde Dateien lesen, einen Port unter 1024 binden, Konten anlegen oder eine Datenbankfunktion ausführen. Meist ist von Privilegien im Plural die Rede, weil ein Subjekt selten nur eines besitzt. „Privilegierte“ und „unprivilegierte“ Konten oder Prozesse bezeichnen die jeweilige Seite dieser Grenze. Das NIST-Glossar ergänzt den leicht übersehenen Teil: Privilegien beschreiben das autorisierte Verhalten eines Subjekts; eine befugte Stelle definiert sie und hält sie in Richtlinien oder Regeln fest.

Neben dem Begriff stehen zwei weitere. Zugriffskontrolle entscheidet über jede einzelne Anfrage. Ein Privileg ist eine der Informationen über den Anfragenden, die diese Entscheidung berücksichtigt. Eine Berechtigung meint meist dieselbe Idee aus Sicht der Ressource: Die Datei legt fest, wer sie lesen darf, und der Leser besitzt das entsprechende Recht.

Unter Linux hat das Wort eine genaue Bedeutung. Die Handbuchseite capabilities(7) beschreibt die traditionelle Unix-Aufteilung: Privilegierte Prozesse laufen mit der effektiven Benutzer-ID 0, heißen root und umgehen alle Berechtigungsprüfungen des Kernels. Jeder andere Prozess wird anhand seiner Identitäts- und Berechtigungsdaten geprüft. Seit Linux 2.2 teilt der Kernel die Macht von root in einzelne, separat ein- und ausschaltbare Einheiten auf, die Capabilities heißen. Ein Programm, das nur einen niedrigen Port binden muss, kann dadurch ausschließlich `CAP_NET_BIND_SERVICE` erhalten. Die Seite credentials(7) erklärt, wessen Identität geprüft wird: Die reale Benutzer-ID sagt, wem ein Prozess gehört; die effektive Benutzer-ID vergleicht der Kernel mit den Berechtigungen. Die gespeicherte Set-User-ID erlaubt einem Programm, eine höher berechtigte Identität vorübergehend abzulegen und später wieder anzunehmen.

Datenbanken und Organisationen verwenden dasselbe Wort für die von ihnen erteilten Rechte. Eine Datenbankrolle besitzt Privilegien auf Schemas, Tabellen und Funktionen. Ein Administratorkonto besitzt Privilegien, die einem Mitarbeiterkonto fehlen. Die Idee ist auf jeder Ebene identisch: ein bewusst gewährtes Recht, eine sonst geschlossene Grenze zu überschreiten.

Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist ein Privileg?“ für eine konkrete Entscheidung. Wir können helfen, das Konzept mit dem System in Verbindung zu bringen, das Sie entwickeln oder einkaufen. Konten, Passwörter und Berechtigungen werden so geordnet, dass alle genau den beruflich erforderlichen Zugriff haben und ihn am Tag ihres Ausscheidens verlieren.

Unsere Leistungen entdecken Kontakt aufnehmen →

Die Annahmen hinter dem Begriff prüfen

Nutzen Sie die Erklärung und die verlinkten Arbeiten, um Fragen für Ihr Projekt zu formulieren. Wir können die Annahmen gemeinsam mit Ihnen prüfen, bevor ein Werkzeug ausgewählt wird. 1.000 RBAC-Regeln für sichere Linux-Server — 1,000 Zugriffsregeln zum Schließen einer Sicherheitslücke

1.000 RBAC-Regeln für sichere Linux-Server Kontakt aufnehmen →

Warum es wichtig ist

Die Privilegien eines Programms bestimmen das Ausmaß des Schadens, den es anrichten kann, wenn etwas mit ihm schiefläuft. Das ist das erklärte Ziel von Capabilities: Ein kompromittiertes Programm mit eng begrenzten Rechten richtet weniger Schaden an als dasselbe Programm unter root. Ein Webserver mit einer Lücke zur Codeausführung und root-Rechten überlässt dem Finder der Lücke den ganzen Rechner. Dieselbe Lücke in einem Prozess, der nur sein eigenes Verzeichnis lesen und einen Port binden darf, reicht viel weniger weit.

NIST hält dies als Prinzip der geringsten Rechte fest. Die Maßnahme AC-6 in SP 800-53 verlangt von einer Organisation, nur jene Zugriffe zu erlauben, die Nutzer und die für sie handelnden Prozesse für ihre zugewiesenen Aufgaben brauchen. Die Ergänzungen machen daraus regelmäßige Arbeit: Administratoren verwenden für Alltagsaufgaben wie E-Mail ein unprivilegiertes Konto, privilegierte Konten bleiben auf die Menschen beschränkt, die sie benötigen, die Nutzung privilegierter Funktionen wird protokolliert, und Privilegien werden geprüft und entzogen, wenn sich ihr Bedarf nicht mehr belegen lässt.

Die Voreinstellungen beginnen selten dort. Bei systemd läuft ein Systemdienst ohne `User=`-Zeile als root. `NoNewPrivileges=`, das verhindert, dass ein Dienst und seine Kindprozesse über `execve()` weitere Rechte erlangen, ist ausgeschaltet, solange es niemand aktiviert. Datenbankwerkzeuge verbinden sich oft als Eigentümer sämtlicher Tabellen, weil das Schema so angelegt wurde. Neue

Mitarbeiterkonten werden häufig von einem bestehenden Konto kopiert, einschließlich aller Rechte. Jede dieser Voreinstellungen ist am ersten Tag bequem und an dem Tag teuer, an dem jemand eindringt.

Der Zielkonflikt ist real. Rechte einzuschränken kostet Zeit, denn jemand muss herausfinden, was ein Programm tatsächlich tut. Zu eng gesetzte Rechte führen erst zur Laufzeit zu Fehlern, nicht schon bei der Installation. Meist entscheidet ein Team daher, wo es diese Zeit zuerst einsetzt: bei allem, was zum Internet hin erreichbar ist, bei allem mit Daten, deren Verlust Menschen treffen würde, und bei den Konten, die andere Konten ändern können. Für Auftraggeber oder Prüfer ist die nützliche Frage einfach und direkt: Wer und was besitzt hier Administratorrechte, und warum?

Wie es in der Praxis aussieht

Auf dem eigenen Host des Unternehmens liefen dreizehn Unit-Dateien mit den Standardprivilegien des Supervisors, also mit den meisten verfügbaren Rechten. Jede Unit-Vorlage erhielt eine auf ihre eigenen Privilegien begrenzte Obergrenze, einen Systemaufruffilter, eingeschränkte Adressfamilien und Speicher, der beschreibbar oder ausführbar ist, aber niemals beides. Der Härtungswert des Totmannschalters sank von 9,6 auf 1,5, der der Git-Forge von 8,3 auf 1,5 und der der beiden Sicherungs-Units von 9,6 auf 2,3 und 2,5. Eine Direktive war in drei Vorlagen falsch geschrieben. Der Supervisor hatte den unbekannten Schlüssel protokolliert und die Unit trotzdem gestartet. Deshalb durchläuft nun jede Unit bei jedem Konvergenzlauf den eigenen Prüfer des Supervisors. Die Seite hält auch fest, was der Wert nicht erfasst: Eine Direktive, die eine Unit funktionsunfähig macht, tut dies erst zur Laufzeit.

Auf einer Anwendungsplattform meldet sich die API mit einer Rolle mit minimalen Rechten an der Datenbank an. Sie darf ausschließlich die Anwendungsfunktionen per EXECUTE aufrufen; diese laufen als SECURITY DEFINER mit parametrisierten Abfragen. Zugriffsberechtigungen wie Tarifstufe, Rolle, Organisation und Begrenzung auf bestimmte Schiffe werden bei jeder Anfrage erneut auf dem Server geprüft. Die Sperren in der Oberfläche gelten lediglich als Teil der Nutzererfahrung (UX).

In einer Organisation wurden Identitäten und Zugriffe über M365, 1Password, Red Hat SSO und OKTA SSO hinweg zusammengeführt, das Prinzip der geringsten Rechte durchgesetzt und Ein- sowie Austrittsprozesse

standardisiert. Dadurch wurden Zugriffe zeitnah entzogen, statt nach einem Weggang weiterzubestehen.

Wo es schiefgeht

Ein Dienst läuft als root, weil seine Unit nichts angibt. Der Supervisor verwendet für Systemdienste standardmäßig root. Eine Unit ohne `User=` -Zeile besitzt daher alle Privilegien des Rechners. Geben Sie jedem Dienst einen eigenen Benutzer oder lassen Sie systemd mit `DynamicUser=` einen zuweisen: Er entsteht beim Start der Unit und wird bei ihrem Ende wieder freigegeben.

Eine Capability verleiht den Großteil der Macht von root. `CAP_SYS_ADMIN` umfasst so viele Operationen, dass die eigene Handbuchseite sie als überladen und als „das neue root“ bezeichnet. Ein Team, das sie für eine einzige Funktion gewährt, hat damit den Großteil der root-Rechte erteilt. Suchen Sie die konkrete Capability, die das Programm braucht, und begrenzen Sie die zulässige Menge darauf:

```
[Service]
User=web
NoNewPrivileges=yes
CapabilityBoundingSet=CAP_NET_BIND_SERVICE
AmbientCapabilities=CAP_NET_BIND_SERVICE
```

Diese Einstellungen erlauben einem als gewöhnlicher Benutzer laufenden Dienst, Port 80 zu binden. Jede andere Capability wird aus der Menge seiner höchstens zulässigen Capabilities entfernt.

Ein Prozess steigt über seine Ausgangsrechte hinaus. Ein Programm unter einem gewöhnlichen Benutzer kann weiterhin eine Set-User-ID-Binärdatei oder eine Datei mit angehängten Capabilities ausführen. Der Kernel erhöht dann seine Privilegien für diesen Lauf. Verteidiger erkennen das an einem Prozess, dessen effektive Benutzer-ID von seiner realen abweicht. `NoNewPrivileges=yes` schließt diesen Weg für einen Dienst und alle von ihm gestarteten Kindprozesse.

Bei Stellenwechseln häufen sich Rechte an. Jemand wechselt das Team und behält den alten Zugriff zusätzlich zum neuen. Nach einigen Jahren besitzt das Konto die vereinigten Rechte aller jemals bekleideten Rollen. AC-6 begegnet

dem mit regelmäßiger Prüfung und AC-2 mit einer an Versetzungen und Austritte gekoppelten Kontoverwaltung. Ein Stellenwechsel wird damit auch zu einer Änderung des Zugriffs.

Die eigene Anmelderolle der Anwendung darf alles. Verbindet sich eine API als Eigentümer des Schemas, reicht eine einzige Injection-Lücke bis zu jeder Tabelle. Eine Anmelderolle, die nur die von der Anwendung genutzten Funktionen aufrufen darf, macht dieselbe Lücke zu einem wesentlich kleineren Problem.

Weiterführende Lektüre

- Privileg im NIST-CSRC-Glossar — die Definition aus CNSSI 4009 mit dem Hinweis, dass eine befugte Stelle Privilegien definiert.
- capabilities(7) — wie Linux root in einzelne Capabilities aufteilt und wie Threads sie tragen.
- credentials(7) — reale, effektive, gespeicherte und Dateisystem-Benutzer-IDs und ihre jeweilige Bedeutung bei Entscheidungen.
- systemd.exec(5) — `User=`, `DynamicUser=`, `NoNewPrivileges=` und die Capability-Einstellungen.
- NIST SP 800-53 Rev. 5 — Maßnahme AC-6, das Prinzip der geringsten Rechte und die Ergänzungen, die es zur Routine machen.

Glossar

Für Ihre tatsächlichen Rahmenbedingungen auswählen

Beschreiben Sie den Bereitstellungsweg, die Betriebsrisiken und die Bereitschaftsverantwortlichen. Wir können eine Änderung mit Prüfungen, einem Wiederherstellungsweg und klaren Zuständigkeiten abgrenzen.

Preise ansehen Kontakt aufnehmen →

Die Begriffe praktisch nutzbar machen

Fragen Sie, wie „Was ist ein Privileg?“ auf Ihre Organisation zutrifft. Ein konkretes Beispiel aus Ihrem bestehenden System reicht aus, um das Gespräch zu beginnen.

Unsere Leistungen entdecken

welcome@engineer.company +45 50 10 61 84 Kopenhagen, Dänemark

Nehmen Sie diese Seite mit

Kontaktkarte

Mit Sorgfalt und Liebe gestaltet von D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – heute · Engineer ApS · Alle Rechte vorbehalten.

Gedruckt von https://engineer.company/de/glossary/privilege/