Engineer

Protokoll

Vereinbarte Regeln für das Gespräch zweier Programme

Was es ist

Ein Protokoll ist eine schriftliche Vereinbarung über ein Gespräch zwischen Programmen: wie jede Nachricht aussieht, was sie bedeutet, welche Seite zuerst spricht, was auf was folgen darf und was jede Seite tut, wenn die andere etwas Unerwartetes sagt. Zwei Programme, von unterschiedlichen Menschen in unterschiedlichen Sprachen geschrieben, können zusammenarbeiten, weil beide nach demselben Dokument gebaut wurden und nicht jeweils nach dem anderen Programm.

Das NIST-Glossar sammelt mehrere Definitionen aus seinen Veröffentlichungen. Ihr gemeinsamer Kern sind Regeln für das Nachrichtenformat und die damit verbundenen Abläufe, denen zwei oder mehr Parteien folgen, um ein gemeinsames Ziel zu erreichen. Eine Definition benennt die leicht vergessene Hälfte: Die Regeln beschreiben neben der Struktur auch die Reihenfolge der Nachrichten. Eine richtig formatierte Nachricht zum falschen Zeitpunkt lässt das Gespräch ebenso sicher scheitern wie eine fehlerhaft formatierte.

Als RFC veröffentlichte Spezifikationen formulieren ihre Regeln mit großgeschriebenen Schlüsselwörtern. MUST ist eine absolute Anforderung. SHOULD erlaubt eine Abweichung nur, wenn ihre vollständigen Auswirkungen verstanden und abgewogen wurden. MAY bezeichnet etwas tatsächlich Optionales. Seit 2017 tragen diese Wörter diese Bedeutung nur in Großbuchstaben; ein kleingeschriebenes „must“ im selben Dokument ist gewöhnliches Englisch. Der Unterschied zwischen SHOULD und MUST ist der zwischen einem ungewöhnlichen und einem fehlerhaften Programm.

Protokolle bilden Schichten. Ein Anwendungsprotokoll, das sich mit Seiten, Menüs oder E-Mail auskennt, setzt meist auf einem Transportprotokoll auf, dessen einzige Aufgabe darin besteht, Bytes zwischen zwei Rechnern zu

bewegen. TCP, der von den meisten verwendete Transport, bietet laut seiner Spezifikation „einen zuverlässigen Dienst für einen geordneten Bytestrom“: Die Bytes kommen vollständig und in der Reihenfolge an, in der sie gesendet wurden. Es verspricht zugleich keinen Zusammenhang zwischen den Abschnitten, die eine Seite schreibt, und denen, die die andere liest. Ein Strom kennt keine Nachrichtengrenzen. Jedes darauf aufbauende Protokoll muss deshalb selbst markieren, wo seine Nachrichten enden: durch ein Zeilenende, eine vorangestellte Länge, das Schließen der Verbindung oder ein anderes vereinbartes Zeichen. UDP trifft die entgegengesetzte Abwägung. Es transportiert getrennte Datagramme mit einem Minimum an Mechanismen und garantiert weder Zustellung noch Schutz vor Duplikaten. Jeder Transport unterscheidet die Dienste eines Rechners anhand einer Portnummer; beide nummerieren ihre Ports getrennt.

Die weiteren Eigenschaften eines Protokolls ergeben sich aus einigen Entscheidungen, die bei jedem Entwurf bewusst oder unbewusst getroffen werden:

- **Wer zuerst spricht.** Manche Server begrüßen den Client, andere warten schweigend auf eine Anfrage. Warten beide Seiten, geschieht nichts, bis ein Zeitgeber abläuft.
- **Was der Server behält.** Bei einem zustandslosen Protokoll trägt jede Anfrage alles für ihre Beantwortung Nötige. Bei einem zustandsbehafteten hängt die Bedeutung einer Nachricht davon ab, was zuvor auf derselben Verbindung geschah.
- **Wie es wächst.** Versionsnummern, Optionscodes und Felder, die der Empfänger ignorieren muss, wenn er sie nicht kennt, lassen einem Protokoll Raum für seine eigene Zukunft.

Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist ein Protokoll?“ für eine konkrete Entscheidung. Wir können helfen, das Konzept mit dem System in Verbindung zu bringen, das Sie entwickeln oder einkaufen. WLAN, Firewalls und Fernzugriff werden so eingerichtet, dass Mitarbeitende von überall arbeiten können und nur Berechtigte Zugang erhalten.

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. 30 Ubuntu-Linux-VPS-Instanzen verwaltet — 30 verwaltete Linux-Server mit geplantem Wiederherstellungsweg

30 Ubuntu-Linux-VPS-Instanzen verwaltet Kontakt aufnehmen →

Warum es wichtig ist

Ein Protokoll macht beide Enden austauschbar. Jeder nach dem Dokument geschriebene Client kann mit jedem danach geschriebenen Server sprechen. Ein Leser kann daher die Software wechseln, und ein Server kann in einer anderen Sprache neu geschrieben werden, ohne dass die andere Seite es bemerkt. Programme, die nur deshalb übereinstimmen, weil sie zusammen gebaut wurden, können sich nicht unabhängig voneinander ändern.

Das Protokoll legt außerdem einen großen Teil der Sicherheitslage fest, bevor überhaupt Code existiert. Welche Seite die Verbindung öffnet, entscheidet darüber, was hereingelassen werden muss: Ein auf Clients wartender Server braucht einen öffentlich erreichbaren Port; ein Programm, das ausschließlich ausgehende Verbindungen aufbaut, braucht keinen. Der Nachrichteninhalt bestimmt, was sich protokollieren, zählen oder preisgeben lässt. Eine Anfrage, die einen Pfad nennt, lässt sich nach Pfad protokollieren. Die Netzwerkadresse eines Lesers zu verbergen hindert den Server nicht daran, Anfragen zu zählen. Ein ohne Verschlüsselung laufendes Protokoll zeigt jede Anfrage jedem durchquerten Netzwerk.

Schließlich überdauern Protokolle die Programme, die sie sprechen. Implementierungen werden alle paar Jahre ersetzt, während das Gespräch über Jahrzehnte weitergeht. Jedes Verhalten, auf das eingesetzte Programme angewiesen sind, wird Teil davon, unabhängig vom Wortlaut des Dokuments. Das Internet Architecture Board hat beide Seiten dieses Problems beschrieben: Fehler, die so lange toleriert werden, bis sie zum Standard werden, und ungenutzten Spielraum für Erweiterungen, der irgendwann nicht mehr funktioniert.

Wie es in der Praxis aussieht

Diese Website antwortet über vier Protokolle aus einem einzigen Dateiordner auf einem Server: HTTP, Gemini, Gopher und einen Onion-Dienst über Tor. Die Seite beschreibt Gopher als Menüprotokoll von 1991 mit unverschlüsseltem TCP, ohne Verschlüsselung und ohne Skripting. Sie zeigt eine vollständige, von Hand eingegebene Anfrage:

```
printf '/contact\r\n' | nc engineer.company 70
```

Sie bezeichnet diese Zeile als das gesamte Protokoll. Darunter schreibt eine Bereitstellung einen Ordner, und vier Eingänge lesen daraus. Keiner davon ist daher eine separat gepflegte Kopie, die auseinanderlaufen könnte.

Firewallregeln werden nach Position statt nach Vorhandensein geprüft, weil ein Paketfilter eine geordnete Liste auswertet und beim ersten Treffer aufhört. Eine Prüfregel betrifft Protokolle: Pro Port gibt es genau eine Regel, immer mit einem Protokoll, denn eine Regel ohne diese Angabe erweitert die Angriffsfläche unbemerkt. Die deklarierte öffentliche Angriffsfläche ist eine kurze Liste von fünf Ports, die als Ganzes geprüft wird. Ein auftauchender, nicht deklarierter Port lässt die Prüfung deshalb scheitern.

Der vierte Eingang ist ein Onion-Spiegel, dessen Protokoll sowohl Firewall als auch Berichterstattung prägte. Der Anonymisierungsdienst erreicht das Netz, indem er ausgehende Tor-Circuits aufbaut: Es wird keine eingehende Verbindung aufgebaut, kein Port wird geöffnet, und keine der beiden Firewallschichten muss etwas synchronisieren. Gemini-Anfragen und Gopher-Verbindungen werden pro Tag ohne Adresse und ohne Pfad gezählt. Der Onion-Dienst hat keinen Zähler und keinen Schalter, um einen hinzuzufügen, weil diese Website diese Leserschaft bewusst ungezählt lässt.

Wo es schiefgeht

Ein tolerierter Fehler wird zum Standard. Das Robustheitsprinzip „Seien Sie konservativ beim Senden und großzügig beim Akzeptieren“ sollte die Zusammenarbeit von Programmen erleichtern. Grenzenlos angewandt lässt es eine fehlerhafte Implementierung in Umlauf kommen und sich verbreiten. Die nächste Implementierung muss denselben Fehler akzeptieren, um mit ihr zusammenzuarbeiten. Er wird zum De-facto-Standard, zu dem alles Neue Fehler für Fehler kompatibel sein muss. RFC 9413 sagt es deutlich: Die eingesetzten Implementierungen werden zu einem Konformitätstest, der die Spezifikation verdrängen kann. Weisen Sie zurück, was die Spezifikation verbietet, möglichst früh im Leben eines Protokolls, und machen Sie den Fehler für jemanden sichtbar, der ihn beheben kann.

Zwei Programme sind uneins, wo eine Nachricht endet. In einem Bytestrom liegt die Grenze dort, wo das Protokoll sie festlegt. Bietet es zwei Arten an, sie zu kennzeichnen, oder akzeptiert ein Parser eine unsaubere Variante, können ein Proxy und der dahinterliegende Server dieselben Bytes als unterschiedliche Nachrichten lesen. Was einer als Ende einer Anfrage behandelt, gilt dem anderen

als neue Anfrage, die der Proxy nie geprüft hat. Lassen Sie eine einzige Regel zur Nachrichtenabgrenzung zu, weisen Sie Mehrdeutiges zurück und lassen Sie jeden Vermittlungsschritt dieselben Regeln zum Parsen anwenden.

Eine Regel nennt einen Port, aber kein Protokoll. Eine Portnummer bedeutet ohne benannten Transport nichts. Je nach Werkzeug kann eine Firewallregel mit einer bloßen Nummer den Port sowohl für TCP als auch für UDP öffnen. Der Dienst, der eines davon brauchte, bekommt es; das andere bleibt für alles offen, was dort später lauscht. Schreiben Sie das Protokoll in jede Regel und vergleichen Sie die gesamte offene Angriffsfläche mit einer deklarierten Liste.

Der Spielraum für Erweiterungen geht verloren. Verwendet kein eingesetzter Sender jemals die Erweiterungspunkte eines Protokolls, entstehen Empfänger, die Unbekanntes zurückweisen. Die erste echte Erweiterung scheitert dann an ihnen. Je länger ein solcher Empfänger im Einsatz bleibt, desto schwerer lässt er sich korrigieren. Testen Sie die Erweiterungspunkte von Anfang an, auch mit bedeutungslosen Werten, damit Toleranz laufend geprüft statt vorausgesetzt wird.

Weiterführende Lektüre

- NIST-CSRC-Glossar: Protokoll — die Definitionen aus den NIST-Sicherheitsveröffentlichungen, jeweils mit ihrem Ursprungsdokument.
- RFC 9293: Transmission Control Protocol — der aktuelle TCP-Standard, einschließlich des Bytestroms ohne Nachrichtengrenzen.
- RFC 768: User Datagram Protocol — das vollständige UDP auf drei Seiten, aus dem Jahr 1980.
- RFC 1436: Das Internet-Gopher-Protokoll — die informatorische Beschreibung eines vollständigen Anwendungsprotokolls mit Beispielsitzungen.
- RFC 9110: HTTP-Semantik — die Bedeutung von Anfragen, Antworten, Methoden und Statuscodes, unabhängig von der übertragenen Version.
- RFC 9413: Robuste Protokolle erhalten — das Internet Architecture Board darüber, warum das Tolerieren von Fehlern sie verfestigt.



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 Protokoll?“ 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/protocol/