1 Netzwerk-Grundlagen

Um Netzwerkprotokolle angreifen zu können, müssen Sie die Grundlagen der Vernetzung von Computern kennen. Je besser Sie verstehen, wie gängige Netzwerke aufgebaut sind und funktionieren, desto einfacher können Sie dieses Wissen nutzen, um neue Protokolle zu erfassen, zu analysieren und auszunutzen.

Im Verlauf dieses Kapitels werde ich grundlegende Konzepte vorstellen, die Ihnen bei der Analyse von Netzwerkprotokollen tagtäglich begegnen. Außerdem schaffe ich auch die Voraussetzung für eine bestimmte Art des Denkens über Netzwerkprotokolle, die es einfacher macht, bisher unbekannte Sicherheitslücken während der Analyse zu entdecken.

1.1 Netzwerkarchitekturen und -protokolle

Wir wollen zuerst einige grundlegende Netzwerkbegriffe besprechen und uns die fundamentale Frage stellen: Was ist ein Netzwerk? Ein *Netzwerk* ist eine Gruppe von zwei oder mehr Computern, die miteinander verbunden sind, um Informationen zu teilen. Jedes mit dem Netzwerk verbundene Gerät wird gewöhnlich als *Knoten* (engl. Node) bezeichnet, um die Beschreibung auf eine größere Palette von Geräten anwenden zu können. Abbildung 1–1 zeigt ein sehr einfaches Beispiel.

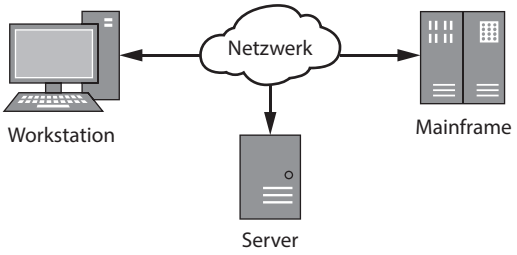

Abb. 1–1 *Einfaches Netzwerk mit drei Knoten*

Die Abbildung zeigt drei Knoten, die über ein gängiges Netzwerk miteinander verbunden sind. Jeder Knoten kann ein anderes Betriebssystem oder eine andere Hardware verwenden. Doch solange jeder Knoten einer Reihe von Regeln folgt, dem *Netzwerkprotokoll*, können sie mit jedem anderen Knoten des Netzwerks kommunizieren. Um sauber miteinander kommunizieren zu können, müssen alle Knoten im Netzwerk das gleiche Netzwerkprotokoll verstehen.

Ein Netzwerkprotokoll übernimmt viele Funktionen, dazu gehören eine oder mehrere der folgenden:

- **Verwaltung des Sessionzustands**
  Protokolle implementieren typischerweise Mechanismen, mit denen neue Verbindungen aufgebaut und vorhandene Verbindungen beendet werden können.
- **Identifizierung von Knoten durch Adressierung**
  Daten müssen im Netzwerk an den richtigen Knoten übertragen werden. Einige Protokolle implementieren einen Adressierungsmechanismus, um bestimmte Knoten oder Gruppen von Knoten zu identifizieren.
- **Flusssteuerung**
  Die Menge der über ein Netzwerk übertragenen Daten ist beschränkt. Protokolle können Wege zur Verwaltung des Datenflusses implementieren, um den Durchsatz zu erhöhen und die Latenz zu reduzieren.
- **Garantierte Reihenfolge der übertragenen Daten**
  Viele Netzwerke garantieren nicht, dass die Reihenfolge, in der die Daten gesendet werden, auch der Reihenfolge entspricht, in der sie eingehen. Ein Protokoll kann die Daten neu ordnen, um die Zustellung in der richtigen Reihenfolge sicherzustellen.
- **Erkennung und Korrektur von Fehlern**
  Viele Netzwerke sind nicht zu 100 Prozent zuverlässig, d.h., Daten können beschädigt werden. Es ist wichtig, Beschädigungen zu erkennen und (idealerweise) zu beheben.
- **Formatierung und Codierung von Daten**
  Daten liegen nicht immer in einem Format vor, das für die Übertragung in einem Netzwerk geeignet ist. Ein Protokoll kann Regeln zur Codierung von Daten festlegen, etwa die Codierung von Text in Binärdaten.

1.2 Die Internet-Protokoll-Suite

TCP/IP ist der von modernen Netzwerken verwendete De-facto-Protokollstandard. Obwohl Sie sich TCP/IP als ein Protokoll vorstellen können, ist es tatsächlich die Kombination von zwei Protokollen: dem *Transmission Control Protocol (TCP)* und dem *Internet Protocol (IP)*. Diese beiden Protokolle sind Teil der *Internet Protocol Suite (IPS)*, eines konzeptionellen Modells, das angibt, wie Netzwerkprotokolle Daten über das Internet senden. Es teilt die Netzwerkkomunikation, wie in Abbildung 1–2 zu sehen, in vier Schichten (Layer) auf.

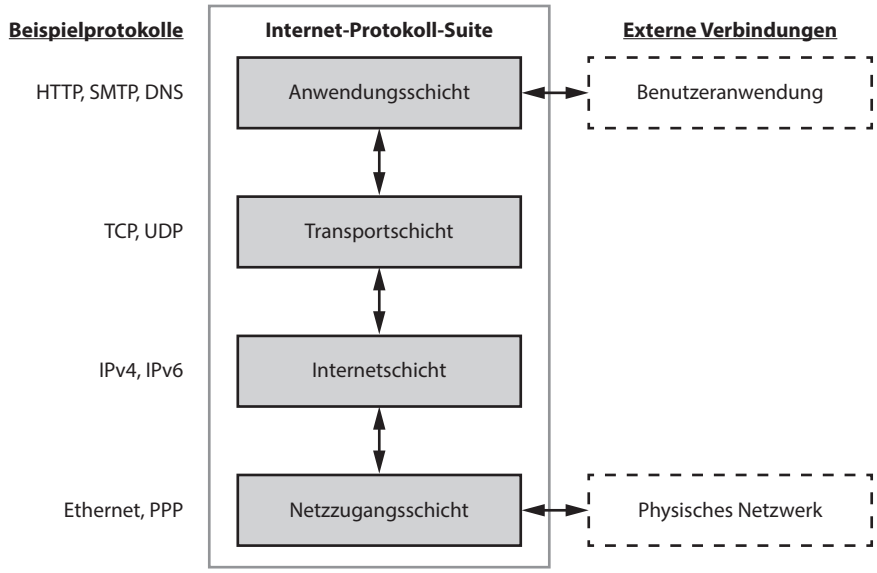

Abb. 1–2 *Schichten der Internet-Protokoll-Suite*

Diese vier Schichten bilden einen *Protokollstack*. Die folgende Liste erläutert jede Schicht der IPS:

- **Netzzugangsschicht (Layer 1)**
  Diese Schicht liegt auf der untersten Ebene und beschreibt die physikalischen Mechanismen, mit denen Informationen zwischen den Knoten eines lokalen Netzwerks übertragen werden. Bekannte Beispiele sind Ethernet (kabelgebunden und drahtlos) und das Point-to-Point-Protokoll (PPP).
- **Internetschicht (Layer 2)**
  Diese Schicht stellt die Mechanismen zur Adressiung der Netzwerkknoten bereit. Im Gegensatz zur Netzzugangsschicht müssen die Knoten nicht im gleichen Netzwerk liegen. Diese Schicht enthält das IP. In modernen Netzwerken kann das verwendete Protokoll die Version 4 (IPv4) oder die Version 6 (IPv6) sein.

- **Transportschicht (Layer 3)**
  Diese Schicht ist für die Verbindungen zwischen Clients und Servern verantwortlich. Manchmal stellt sie auch die korrekte Reihenfolge der Pakete sicher oder bietet das Multiplexing von Diensten an. Das Multiplexing von Diensten erlaubt es einem einzelnen Knoten, unterschiedliche Dienste zu unterstützen, indem jedem Service eine andere Nummer zugewiesen wird. Diese Nummer wird als *Port* bezeichnet. TCP und UDP (User Datagram Protocol) arbeiten auf dieser Schicht.
- **Anwendungsschicht (Layer 4)**
  Auf dieser Schicht sind Netzwerkprotokolle wie das *HyperText Transport Protocol (HTTP*, zur Übertragung von Webseiten), das *Simple Mail Transport Protocol (SMTP*, zur Übertragung von E-Mails) und das *Domain Name System (DNS,* zur Umwandlung von Namen in Adressen) angesiedelt. In diesem Buch konzentrieren wir uns hauptsächlich auf diese Schicht.

Jede Schicht interagiert nur mit der direkt über oder unter ihr liegenden Schicht, doch es muss auch externe Interaktionen mit dem Stack geben. Abbildung 1–2 zeigt zwei externe Verbindungen. Die Netzzugangsschicht interagiert mit einer physikalischen Netzwerkverbindung und überträgt Daten in ein physikalisches Medium wie Strom- oder Lichtimpulse. Die Anwendungsschicht interagiert mit der Anwendung: Eine *Anwendung* ist eine Sammlung zusammengehöriger Funktionalitäten, die dem Benutzer einen Dienst zur Verfügung stellen. Abbildung 1–3 zeigt beispielhaft eine Anwendung zur Verarbeitung von E-Mails. Der Dienst, der von der E-Mail-Anwendung angeboten wird, ist das Senden und Empfangen von Nachrichten über ein Netzwerk.

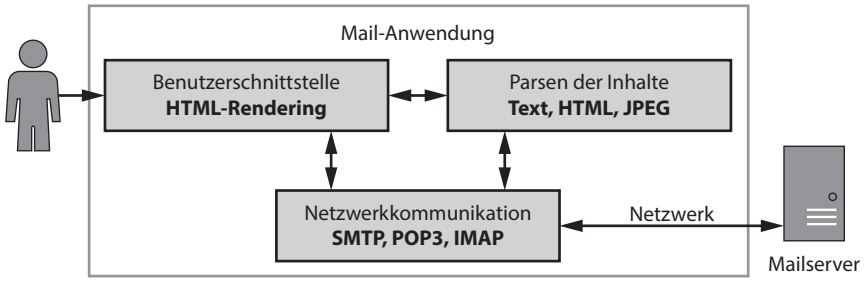

Abb. 1–3 *Beispielhafte E-Mail-Anwendung*

Typischerweise umfassen Anwendungen die folgenden Komponenten:

- **Netzwerkkommunikation**
  Diese Komponente kommuniziert über das Netzwerk und verarbeitet ein- und ausgehende Daten. Bei einer E-Mail-Anwendung läuft die Netzwerkkommunikation meist über ein Standardprotokoll wie SMTP oder POP3.
- **Parsen der Inhalte**
  Über ein Netzwerk transferierte Daten müssen üblicherweise extrahiert und verarbeitet werden (Parsen). Bei den Inhalten kann es sich um Textdaten (z.B. den Text der E-Mail), Bilder oder Videos handeln.
- **Benutzerschnittstelle**
  Die Benutzerschnittstelle (User Interface, kurz UI) erlaubt es dem Benutzer, empfangene E-Mails anzusehen und neue E-Mails zu verfassen bzw. zu senden. Bei einer E-Mail-Anwendung könnte die UI E-Mails mittels HTML in einem Webbrowser darstellen.

Beachten Sie, dass der Benutzer, der mit der UI interagiert, kein Mensch sein muss. Es kann sich auch um eine andere Anwendung handeln, die das Senden und Empfangen von E-Mails (z.B. über ein Kommandozeilen-Tool) automatisiert.

1.3 Datenkapselung

Jede Schicht der IPS baut auf der darunterliegenden Schicht auf und jede Schicht ist in der Lage, Daten der darüberliegenden Schicht zu kapseln, sodass sie zwischen den Schichten bewegt werden können. Die von jeder Schicht übertragenen Daten werden *Protocol Data Unit (PDU)* genannt.

1.3.1 Header, Footer und Adressen

Die PDU jeder Schicht enthält die zu übertragenden Nutzdaten. Üblicherweise stellt man den Nutzdaten einen *Header* voran, der für die Übertragung der Nutzdaten benötigte Informationen enthält, wie z.B. die *Adressen* der Quell- und Zielknoten. Manchmal besitzt eine PDU auch einen *Footer,* der an die Nutzdaten angehängt wird und Werte enthält, die eine korrekte Übertragung sicherstellen, etwa Prüfsummen. Abbildung 1–4 zeigt, wie die PDUs der IPS ausgelegt sind.

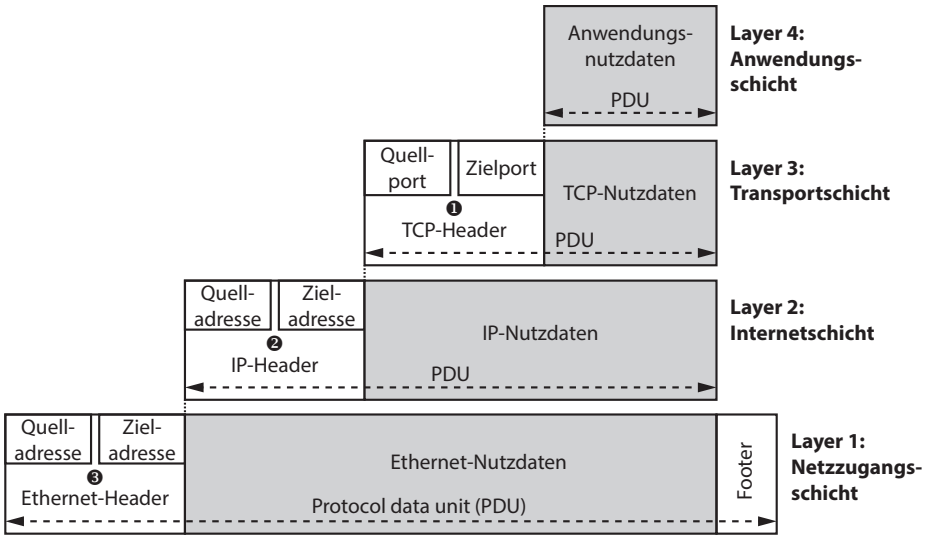

Abb. 1–4 *IPS-Datenkapselung*

Der TCP-Header enthält den Quell- und Zielport ❶. Diese Portnummern erlauben es einem einzelnen Knoten, mehrere Netzverbindungen aufzubauen. Die Portnummern für TCP (und UDP) reichen von 0 bis 65535. Die meisten Portnummern werden neuen Verbindungen nach Bedarf zugewiesen, doch einigen Nummern wurden bestimmte Dienste zugeordnet, wie etwa Port 80 für HTTP. (Eine aktuelle Liste der zugewiesenen Portnummern finden Sie bei den meisten unixoiden Betriebssystemen in der Datei */etc/services*.) Die TCP-Nutzdaten und den Header nennt man üblicherweise ein *Segment*, während UDP-Nutzdaten und -Header als *Datagramm* bezeichnet werden.

Das IP-Protokoll verwendet eine Quell- und eine Zieladresse ❷. Die *Zieladresse* erlaubt das Senden der Daten an einen bestimmten Knoten im Netzwerk. Über die *Quelladresse* weiß der Empfänger, welcher Knoten ihm Daten gesendet hat, was es ihm ermöglicht, dem Sender zu antworten.

IPv4 verwendet 32-Bit-Adressen, die üblicherweise als vier durch Punkte getrennte Zahlen wie z.B. 192.168.10.1 dargestellt werden. IPv6 nutzt 128-Bit-Adressen, weil 32-Bit-Adressen für die Anzahl der Knoten in modernen Netzwerken nicht mehr ausreichen. IPv6-Adressen werden üblicherweise als durch Doppelpunkte getrennte Hexadezimalzahlen wie z.B. fe80:0000:0000:0000:897b:581e:44b0:2057 geschrieben. Lange Folgen von 0000-Werten werden durch zwei Doppelpunkte ersetzt, d.h., die obige IPv6-Adresse kann auch als fe80::897b:581e:44b0:2057 geschrieben werden. IP-Nutzdaten und -Header werden üblicherweise als Paket (*packet)* bezeichnet.

Ethernet enthält ebenfalls Quell- und Zieladressen ❸. Ethernet verwendet einen als MAC-Adresse (*Media Access Control)* bezeichneten 64-Bit-Wert, der normalerweise bei der Produktion des Ethernet-Adapters festgelegt wird. MAC-

Adressen werden üblicherweise als Folge von durch Minuszeichen oder Doppelpunkte getrennten Hexadezimalzahlen wie 0A-00-27-00-00-0E dargestellt. Die Ethernet-Nutzdaten, zusammen mit dem Header und dem Footer, werden üblicherweise als *Frame* bezeichnet.

1.3.2 Datenübertragung

Sehen wir uns kurz an, wie beim IPS-Modell der Datenkapselung Daten von einem Knoten zum anderen übertragen werden. Abbildung 1–5 zeigt ein einfaches Ethernet-Netzwerk mit drei Knoten.

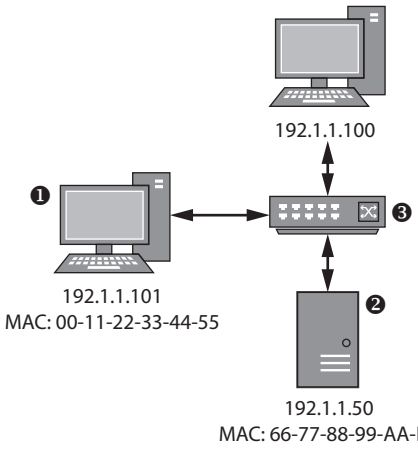

Abb. 1–5 *Einfaches Ethernet-Netzwerk*

In diesem Beispiel möchte der Knoten ❶ mit der IP-Adresse 192.1.1.101 Daten per IP-Protokoll an den Knoten ❷ mit der IP-Adresse 192.1.1.50 senden. (Der *Switch* ❸ leitet Ethernet-Frames zwischen allen Netzwerkknoten weiter. (Der Switch benötigt keine IP-Adresse, da er nur auf der Netzzugangsschicht arbeitet.) Wenn Daten zwischen den beiden Knoten gesendet werden sollen, passiert Folgendes:

1. Der Netzwerkstack des Betriebssystems ❶ kapselt die Daten der Anwendungs- und der Transportschicht und erzeugt ein IP-Paket mit der Quelladresse 192.1.1.101 und der Zieladresse 192.1.1.50.
2. An diesem Punkt kann das Betriebssystem die IP-Daten in einem Ethernet-Frame kapseln, kennt möglicherweise aber nicht die MAC-Adresse des Zielknotens. Es kann die MAC-Adresse für eine bestimmte IP-Adresse über das Address Resolution Protocol (ARP) anfordern, das einen Request an alle Knoten im Netzwerk sendet, um die MAC-Adresse für die IP-Adresse des Ziels zu ermitteln.

3. Sobald der Knoten an ❶ die ARP-Antwort erhält, kann er den Frame erzeugen, wobei die Quelladresse mit der lokalen MAC-Adresse 00-11-22-33-44-55 und die Zieladresse mit 66-77-88-99-AA-BB angegeben wird. Der neue Frame wird in das Netzwerk übertragen und vom Switch ❸ empfangen.
4. Der Switch leitet den Frame an den Zielknoten weiter, der das IP-Paket entpackt und prüft, ob die Ziel-IP-Adresse stimmt. Dann werden die IP-Nutzdaten entpackt und im Stack weitergeleitet, um von der wartenden Anwendung empfangen zu werden.

1.4 Netzwerk-Routing

Ethernet verlangt, dass alle Knoten direkt mit dem gleichen Netzwerk verbunden sind. Diese Anforderung ist für ein wirklich globales Netzwerk eine wesentliche Einschränkung, da es praktisch unmöglich ist, physikalisch jeden Knoten mit jedem anderen Knoten zu verbinden. Statt also alle Knoten direkt miteinander verbinden zu müssen, erlauben es die Quell- und Zieladressen, dass Daten über verschiedene Netzwerke *gerouted* (weitergeleitet) werden, bis sie den gewünschten Zielknoten erreichen (siehe Abb. 1–6).

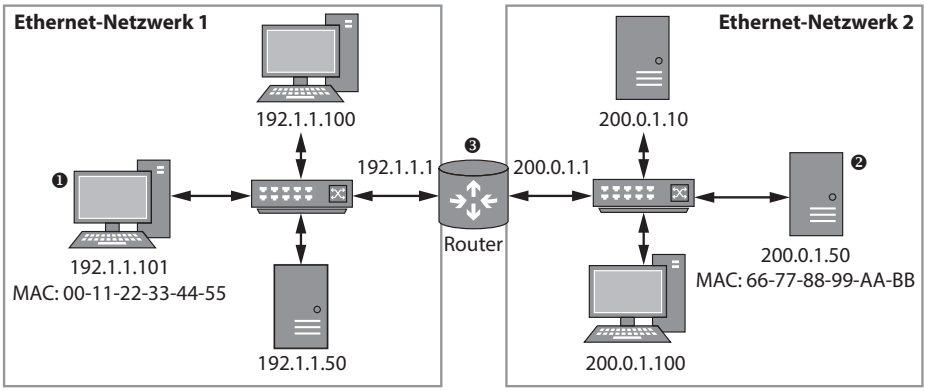

Abb. 1–6 *Routing zwischen zwei Ethernet-Netzwerken*

Abbildung 1–6 zeigt zwei Ethernet-Netzwerke mit eigenen IP-Adressbereichen. Die folgende Beschreibung erklärt, wie das IP-Protokoll dieses Modell nutzt, um Daten von dem Knoten bei ❶ in Netzwerk 1 an den Knoten bei ❷ in Netzwerk 2 zu senden.

1. Der Netzwerkstack des Betriebssystems auf dem Knoten bei ❶ kapselt die Daten der Anwendungs- und Transportschicht und baut ein IP-Paket mit der Quelladresse 192.1.1.101 und der Zieladresse 200.0.1.50 auf.
2. Der Netzwerkstack muss einen Ethernet-Frame senden, da aber die IP-Zieladresse in keinem Ethernet-Netzwerk existiert, mit dem der Knoten verbunden

ist, befragt der Netzwerkstack die *Routing-Tabelle* des Betriebssystems. In diesem Beispiel enthält die Routing-Tabelle einen Eintrag für die IP-Adresse 200.0.1.50. Dieser Eintrag zeigt an, dass der Router ❸ an IP-Adresse 192.1.1.1 weiß, wie man zu dieser Zieladresse gelangt.

3. Das Betriebssystem nutzt ARP, um die MAC-Adresse des Routers an 192.1.1.1 nachzuschlagen und das Original-IP-Paket wird im Ethernet-Frame mit dieser MAC-Adresse gekapselt.
4. Der Router empfängt den Ethernet-Frame und entpackt das IP-Paket. Wenn er die IP-Zieladresse prüft, erkennt er, dass dieses IP-Paket nicht für diesen Router, sondern für einen Knoten in einem anderen Netzwerk gedacht ist. Der Router schlägt die MAC-Adresse für 200.0.1.50 nach, kapselt das ursprüngliche IP-Paket in einem neuen Ethernet-Frame und sendet diesen an Netzwerk 2.
5. Der Zielknoten empfängt den Ethernet-Frame, entpackt das IP-Paket und verarbeitet dessen Inhalt.

Dieser Routing-Prozess kann sich mehrfach wiederholen. Ist der Router beispielsweise nicht direkt mit dem Netzwerk verbunden, das den Knoten 200.0.1.50 enthält, würde er seine eigene Routing-Tabelle konsultieren und den nächsten Router bestimmen, an den er das IP-Paket sendet.

Natürlich ist es praktisch nicht möglich, dass jeder Knoten im Netzwerk weiß, wie er zu jedem anderen Knoten im Internet gelangt. Wenn es für ein Ziel keinen expliziten Routing-Eintrag gibt, stellt die Routing-Tabelle einen Standardeintrag bereit, das sogenannte *Default Gateway*, der die IP-Adresse eines Routers enthält, der IP-Pakete an ihr Ziel weiterleiten kann.

1.5 Mein Modell für die Analyse von Netzwerkprotokollen

Die IPS beschreibt, wie die Netzwerkkommunikation funktioniert. Für Analysezwecke ist ein Großteil des IPS-Modells aber nicht relevant. Es ist einfacher, mein Modell zu nutzen, um das Verhalten eines Anwendungsprotokolls zu verstehen. Mein Modell besteht aus drei Schichten. Abbildung 1–7 zeigt diese Schichten und verdeutlicht, wie ich einen HTTP-Request analysieren würde.

Hier die drei Schichten meines Modells:

- **Inhaltsschicht (Content Layer)**
  Gibt den »Sinn« dessen wieder, was kommuniziert wird. In Abbildung 1–7 besteht der Sinn darin, mit einem HTTP-Request die Datei *image.jpg* abzurufen.
- **Codierungsschicht (Encoding Layer)**
  Legt die Regeln fest, nach denen der Inhalt repräsentiert werden soll. In diesem Beispiel wird die HTTP-Anfrage als HTTP-GET-Request codiert, der die abzurufende Datei festlegt.

- **Transportschicht (Transport Layer)**
  Legt die Regeln fest, nach denen die Daten zwischen den Knoten übertragen werden. In diesem Beispiel wird der HTTP-GET-Request über eine TCP/IP-Verbindung mit Port 80 des entfernten Knotens durchgeführt.

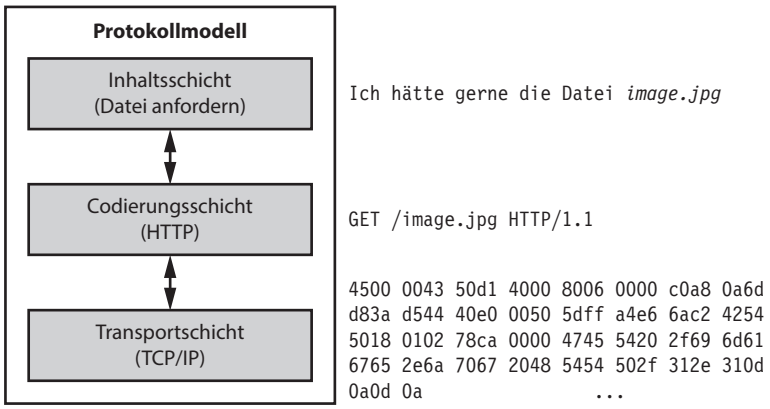

Abb. 1–7 *Mein Modellkonzept für Protokolle*

Diese Art der Aufteilung des Modells reduziert die Komplexität anwendungsspezifischer Protokolle, weil wir die Teile des Netzwerkprotokolls herausfiltern können, die für uns nicht relevant sind. Da es uns beispielsweise nicht interessiert, wie TCP/IP an den entfernten Knoten gesendet wird (wir gehen einfach davon aus, dass es irgendwie funktioniert), können wir TCP/IP-Daten als binären Transport betrachten, der schlicht funktioniert.

Um zu verstehen, warum dieses Protokollmodell nützlich ist, stellen Sie sich einfach vor, Sie müssen den Netzwerkverkehr irgendeiner Malware untersuchen. Sie finden heraus, dass die Malware HTTP nutzt, um Befehle vom Operator über einen Server zu empfangen. Der Operator könnte die Malware zum Beispiel anweisen, alle Dateien auf der Festplatte des infizierten Computers aufzulisten. Die Dateiliste kann an den Server zurückgeschickt werden und der Operator kann dann den Upload einer bestimmten Datei anfordern.

Wenn wir das Protokoll aus dem Blickwinkel betrachten, wie der Operator mit der Malware interagiert, indem er z.B. den Upload einer Datei veranlasst, können wir das neue Protokoll in die in Abbildung 1–8 aufgeführten Schichten aufteilen.

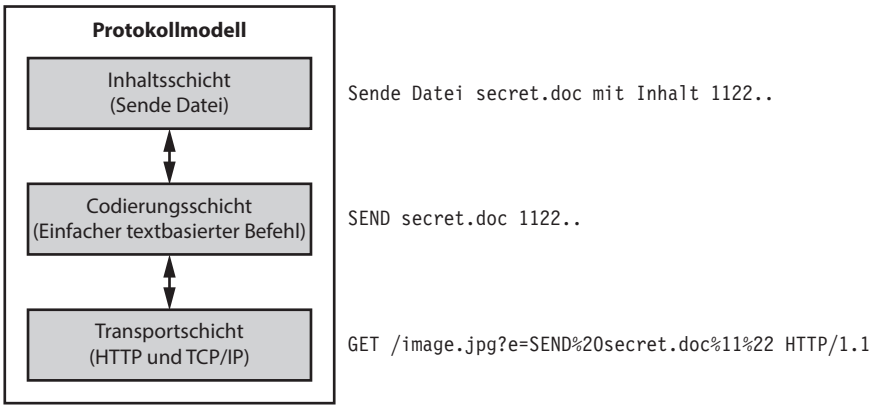

Abb. 1–8 *Modellkonzept für ein HTTP nutzendes Malware-Protokoll*

Die folgende Liste erläutert jede Schicht des neuen Protokollmodells:

- **Inhaltsschicht**
  Die bösartige Anwendung sendet eine gestohlene Datei namens *secret.doc* an den Server.
- **Codierungsschicht**
  Die Codierung des Befehls zum Senden der gestohlenen Datei besteht aus einem einfachen Textstring mit dem Befehl `SEND`, gefolgt vom Dateinamen und den Daten.
- **Transportschicht**
  Das Protokoll verwendet einen HTTP-Request-Parameter, um den Befehl zu übertragen. Es benutzt die übliche Prozentcodierung, um einen gültigen HTTP-Request zu erzeugen.

Beachten Sie, dass wir in diesem Beispiel nicht berücksichtigt haben, wie der HTTP-Request über TCP/IP gesendet wird. Wir haben die Codierungs- und Transportschicht aus Abbildung 1–7 in Abbildung 1–8 in der Transportschicht zusammengefasst. Zwar nutzt die Malware Low-Level-Protokolle wie TCP/IP, doch diese Protokolle sind nicht wichtig, wenn wir analysieren wollen, wie der Malware-Befehl eine Datei sendet. Das ist deshalb nicht wichtig, weil wir HTTP über TCP/IP als einzelne Transportschicht betrachten können, die einfach funktioniert, und uns lieber auf die Malware-Befehle konzentrieren wollen.

Indem wir unseren Blick auf die Schichten des Protokolls richten, die wir analysieren müssen, vermeiden wir viel Arbeit und können uns auf die wesentlichen Aspekte des Protokolls konzentrieren. Würden wir dieses Protokoll andererseits nach den Schichten aus Abbildung 1–7 analysieren, könnten wir annehmen, dass die Malware einfach die Datei *image.jpg* anfordert, weil es so aussieht, als wäre das alles, was der HTTP-Request macht.

1.6 Am Ende dieses Kapitels

Dieses Kapitel hat kurz in die Netzwerk-Grundlagen eingeführt. Ich habe die IPS vorgestellt sowie einige der Protokolle, denen Sie in echten Netzwerken begegnen werden. Außerdem habe ich gezeigt, wie Daten zwischen Knoten eines lokalen Netzwerks und über Router auch an entfernte Netzwerke übertragen werden. Darüber hinaus habe ich einen Weg beschrieben, Anwendungsprotokolle zu betrachten, der es Ihnen einfacher macht, sich auf die spezifischen Features des Protokolls zu konzentrieren und so die Analyse zu beschleunigen.

In Kapitel 2 werden wir diese Netzwerk-Grundlagen nutzen, um den Netzwerkverkehr für die Analyse zu erfassen, was man als Capturing bezeichnet. Das Ziel des Erfassens von Netzwerkverkehr besteht darin, auf die Daten zugreifen zu können, die Sie benötigen, um mit dem Analyseprozess zu beginnen, die verwendeten Protokolle zu identifizieren und letztlich die Sicherheitslücken aufzuspüren, die Sie ausnutzen können, um Anwendungen zu kompromittieren, die dieses Protokoll verwenden.