Kapazitätsmessung eines verdeckten Zeitkanals über HTTP

Hans-Georg Eßer Felix C. Freiling
Lehrstuhl für Praktische Informatik I, Universität Mannheim

Abstract: Wir beschreiben die Implementierung eines einfachen verdeckten Zeitkanals über HTTP und evaluieren dessen Kapazität im Internet. Im Experiment kommunizierte ein modifizierter Apache-Webserver mit einem selbst geschriebenen HTTP-Proxy auf der Seite des Clients. Optimiert man den Kanal auf Fehlerfreiheit, können 3 Bit/s übertragen werden; akzeptiert man bis zu 10 % Fehler, sind 14 Bit/s möglich. Die einfache Machbarkeit demonstriert erneut die Gefährlichkeit verdeckter Kanäle.

1 Einleitung

Es ist seit langem bekannt, dass selbst ein perfekter Zugriffsschutzmechanismus keine vollständige Vertraulichkeit erreichen kann. Die Ursache hierfür sind die so genannten *verdeckten Kanäle*. Ein Kommunikationskanal gilt als verdeckt, wenn er ursprünglich nicht für Kommunikation eingerichtet wurde. Der Begriff des verdeckten Kanals geht auf eine Arbeit von Lampson [Lam73] zurück. Man unterscheidet zwischen *Ressourcen-* und *Zeitkanälen*. Während erstere durch den gemeinsamen Zugriff der beteiligten Prozesse auf eine Systemressource (wie Plattenplatz, Rechenzeit, Hauptspeicher) geprägt sind, arbeiten Zeitkanäle mit manipulierten zeitlichen Abständen zwischen zwei Systemereignissen.

Verdeckte Kanäle stellen eines der am schwersten beherrschbaren Phänomene in der IT-Sicherheit dar. Während frühere Arbeiten die Gefahren im Wesentlichen im (militärischen) Hochsicherheitsbereich sahen [Eck03], ist heute aufgrund der steigenden Verbreitung von Viren, Würmern und Trojanischen Pferden die Gefährdung von Firmen und Privatpersonen unabstreitbar. In der industriellen Praxis wird versucht, verdeckte Kanäle durch Designmethoden [Kem83] oder komplexe Schleusentechnologien zu erkennen und zu vermeiden. In privaten Netzen verhindern Techniken wie *Network Address Translation* (NAT) oder eine Firewall ungewollten Datenverkehr. Wie diese Arbeit zeigt, bleiben die Gefahren durch verdeckte Kanäle dort jedoch grundsätzlich bestehen.

Diese Arbeit basiert auf folgendem Szenario: Ein Nutzer unterhält einen abgesicherten Rechner mit vertraulichen Daten. Auf dem Rechner läuft als einziger Dienst ein Webserver, der einen gewöhnlichen HTTP-Zugang über den TCP-Netzwerkport 80 anbietet. Auf dem Server wurde jedoch ein Trojanisches Pferd platziert, welches mit einem Nutzer im Internet kommunizieren möchte. Als Kommunikationskanal steht ausschließlich ein verdeckter Zeitkanal über Port 80 zur Verfügung, der eigentliche Inhalt der Verbindung bleibt dabei unverändert.

Die Ausgangsfrage dieser Arbeitet lautete: Wie einfach ist es möglich, einen verdeckten Zeitkanal mit Standardmethoden zu implementieren und welche Kapazität hat er? Hierfür wurde eine HTTP-Verbindung zwischen einem standardmäßigen Apache-Webserver und

einem HTTP-Proxy als Trägerkanal für einen Zeitkanal verwendet. Der Webserver verzögert beim Übertragen einer größeren Datei einzelne Blöcke auf der TCP-Verbindung abhängig davon, ob auf dem verdeckten Kanal der Wert 0 oder 1 übertragen werden soll. Die auf diese Weise maximal erreichbare relative Kapazität ist offensichtlich 1 Bit pro Block. Die Blockgröße, die vom Apache-Webserver verwendet wird, beträgt 4 KByte.

Die Maximalkapazität des hier vorgestellten Zeitkanals hängt direkt von der gewählten Verzögerungszeit einzelner Bits ab und schwankt in den Messungen zwischen 4 und 28 Bit/s; für Fehlerraten unter 5 % liegt die Kapazität zwischen 4 und 9 Bit/s. Die hier präsentierten Ergebnisse sind zusammen mit einigen weiteren Messungen an anderem Ort ausführlicher dokumentiert [Eße05]; eine Langfassung dieser Arbeit ist zudem als technischer Bericht verfügbar [EF05].

2 Experiment

Der Aufbau der Experiments ist in Abbildung 1 dargestellt. Der Rechner des „Spions“ ist ganz links abgebildet. Er kommuniziert über die HTTP-Verbindung mit dem Apache-Webserver (Mitte). Die Software-Architektur auf dem Webserver ist rechts skizziert. Um den verdeckten Kanal zu erzeugen, wurden drei Komponenten eingesetzt: (1) Apache-Server, (2) Kontroll-Daemon „cc daemon“ und (3) HTTP-Proxy (der auf dem Rechner ganz links läuft). Das Modul „ccctrl“ ist das eigentliche Trojanische Pferd, welches über den Kontroll-Daemon den Inhalt der versendeten Nachrichten steuern kann. In der vorliegenden Implementierung kann es über zahlreiche Kommandozeilenparameter gesteuert werden [Eße05].

Die Kommunikation läuft wie folgt ab: Zunächst stellt ein Webclient über den HTTP-Proxy eine Anfrage an den Webserver. Der Webserver fragt bei jedem Zugriff beim Kontroll-Daemon an, ob eine verdeckte Nachricht gesendet werden soll. Letzterer prüft in seiner Datenbank, ob die IP-Adresse in der Liste der geheimen Kommunikationspartner enthalten ist. Falls ja, schickt er das nächste Bit einer für diese Adresse gespeicherten verdeckten Nachricht als 0 oder 1 kodiert an den Apache-Server. Abhängig von dieser Antwort verschickt der Apache-Server den aktuellen Block verzögert (1) oder unverzögert (0). Der HTTP-Proxy misst die Übertragungszeit (die Zeit zwischen dem Aufruf von `read()` und dem Empfang des Blocks) und protokolliert diese Zeiten. Außerdem reicht er den eigentlichen Inhalt (die Daten des Trägerkanals) an den Webclient weiter. Nachdem die

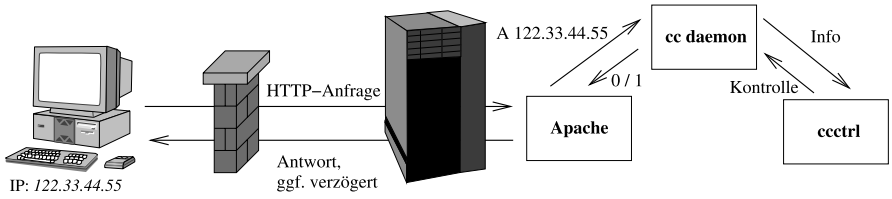

Abbildung 1: Kommunikation zwischen Client und Server sowie lokal auf dem Server.

Datei komplett übertragen wurde, wird die Protokolldatei des HTTP-Proxys mit statistischen Methoden analysiert, um die verdeckte Nachricht zu rekonstruieren.

Für die Tests wurden zwei Rechner verwendet: Der Server (Intel Celeron, 2 GHz, 256 MByte RAM, Debian Linux 3.0) steht im Rechenzentrum eines Hosting-Anbieters, der Client (Intel Pentium IV, 2 GHz, 512 MByte RAM, Suse Linux 9.0) ist über eine DSL-Leitung (768 MBit/s *up-stream*, 128 MBit/s *down-stream*) mit dem Internet verbunden. Für 15 Verzögerungswerte zwischen 0,05 s und 0,5 s wurden je fünf Testläufe ausgeführt, bei denen eine 5 MByte große Testdatei auf dem Trägerkanal übertragen wurde. Die vollständig unverzögerte Übertragung der Testdatei benötigte über diese Verbindung 42 s; bei Verzögerung von durchschnittlich jedem zweiten 4-KByte-Block um 0,5 s erhöhte sich die Übertragungszeit auf 322 s, was im Rahmen der Erwartungen liegt.

3 Ergebnisse

Gemessen wurden die Fehlerraten bei verschiedenen Verzögerungszeiten. Es war zu erwarten, dass kürzere Verzögerungen zu mehr Fehlern führen. Abbildung 2 zeigt die gemessenen Übertragungszeiten für zwei Verzögerungen v. Auf der horizontalen Achse sind die Nummern der 4-KByte-Blöcke aufgezeichnet. Bei einer Verzögerung von $v = 0,4$ s war die Übertragung zu 0,64 % fehlerbehaftet (siehe Abbildung 2 links). Eine deutliche Trennung verzögerter und unverzögerter Pakete in zwei Messwertblöcke ist optisch nicht erkennbar. Eine Verzögerung von $v = 0,05$ s führte zur völligen Unbrauchbarkeit des Kanals: Mit 50,64 % Fehlerrate war der Kanal vollständig gestört. Abbildung 2 (rechts) zeigt, dass sich fast alle Messwerte in einem kleinen Intervall um 0,0033 s befinden – die erwartete Aufteilung in zwei Blöcke gab es nicht.

Abbildung 3 zeigt die Fehlerquoten für die verschiedenen Verzögerungszeiten im linken Graphen mit linearer y-Achse und im rechten Graphen mit logarithmischer y-Achse. Zum Vergleich wurde in beide Graphen die Exponentialfunktion hineingelegt. Man kann hier feststellen, dass die Fehlerquote exponentiell steigt, wenn die Verzögerung reduziert wird. Für $v = 0,05$ s (und kleinere v) lag die Fehlerrate bei 50 % – hier war keine Unterscheidung zwischen verzögerten und unverzögerten Blöcken mehr möglich, und das Analyse-

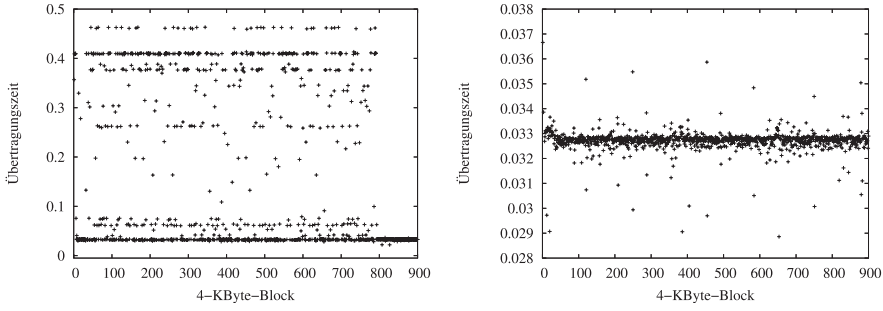

Abbildung 2: Messungen mit $v = 0,4$ s (links) und $v = 0,05$ s (rechts).

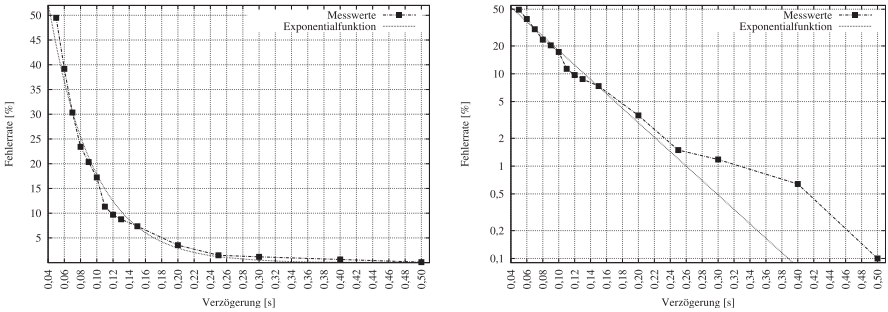

Abbildung 3: Zusammenhang zwischen Verzögerung und Fehlerrate (links: lineare y-Achse, rechts: logarithmische y-Achse; Verbindungslinien nur zur besseren Veranschaulichung).

programm auf der Empfängerseite interpretierte alle Blöcke gleich. Da die Nullen und Einsen in der verdeckten Nachricht annähernd gleich verteilt sind, ist 50 % das schlechtestmögliche Ergebnis. Bei den Tests mit 0,06 s $\leq v \leq$0,15 s wurden 39,18 % bis 7,36 % der Bits fehlerhaft übertragen. Ab $v = 0,2$ s lag die Fehlerquote unter 3,54 %. Mit einfachen Fehlerkorrekturverfahren (Hamming Codes) konnte dies jedoch nur leicht verbessert werden [EF05]. Selbst bei einer halben Sekunde Verzögerung traten noch Fehler auf – im Mittel waren 0,1 % der Bits falsch.

4 Schlussfolgerungen

Die Untersuchung hat gezeigt, dass vorhandene Software mit geringem Aufwand um einen verdeckten Kanal ergänzt werden kann, der keine inhaltlichen Veränderungen an den übertragenen Daten vornimmt und somit durch eine Überwachung dieser Daten nicht aufzufinden ist – das ist lediglich durch Beobachtung der zeitlichen Auffälligkeiten möglich. Die Machbarkeit – und damit auch die Gefährlichkeit – verdeckter Kanäle wurde damit experimentell nachgewiesen.

Literatur

[Eck03] Claudia Eckert. *IT-Sicherheit*. Oldenbourg Wissenschaftsverlag, 2. Auflage, 2003.

[EF05] Hans-Georg Eßer und Felix C. Freiling. Kapazitätsmessung eines verdeckten Zeitkanals über HTTP. Bericht TR-2005-10, Universität Mannheim, Fakultät für Mathematik und Informatik, 2005.

[Eße05] Hans-Georg Eßer. *Ausnutzung verdeckter Kanäle am Beispiel eines Web-Servers*. Diplomarbeit, RWTH Aachen, Mai 2005. `http://privat.hgesser.com/docs/`.

[Kem83] Richard A. Kemmerer. Share resource matrix methodology: An approach to identifying storage and timing channels. *ACM Transactions on Computer Systems*, 1(3):256–277, August 1983.

[Lam73] Butler W. Lampson. A note on the confinement problem. *Commun. ACM*, 16(10):613–615, 1973.