Magdeburger Journal zur Sicherheitsforschung

Gegründet 2011 | ISSN: 2192-4260
Herausgegeben von Stefan Schumacher und Jan W. Meine
Meine Verlag Magdeburg

Einführung in die Forschungsthematik der verdeckten Kanäle*

Steffen Wendzel, Jörg Keller

Verdeckte Kanäle stellen eine bisher nur wenig außerhalb der Forschung betrachtete Technik für unzensierte Kommunikation dar. Zugleich liegt in verdeckten Kanälen die Gefahr des unbemerkten Informationsverlusts (Data Leakage) und der unbemerkten Steuerung von Botnetzen. Dieser Artikel bietet zunächst eine Einführung in die Thematik der verdeckten Kanäle, ihre Risiken und Chancen. Anschließend werden exemplarisch einzelne Techniken zur Erzeugung von verdeckten Kanälen vorgestellt, sowie gängige Gegenmaßnahmen diskutiert. Der Artikel schließt mit einem Überblick über neuere Techniken der letzten Jahre.

Schlüsselwörter: Verdeckter Kanal, Netzwerksicherheit, TCP/IP

**Der Artikel erscheint in der Serie zum Datenspuren-Symposium des Chaos Computer Club Dresden. 2011*

Zitationsvorschlag: Wendzel, Steffen / Keller, Jörg (2011). Einführung in die Forschungsthematik der verdeckten Kanäle. Magdeburger Journal zur Sicherheitsforschung, Band 2, 2011, S. 115–124.

http://www.wissens-werk.de/index.php/mjs/article/viewFile/108/78

Einführung

Der Begriff des verdeckten Kanals (*covert channels*) wurde bereits von Lampson (1973) geschaffen und bezeichnet zunächst einen Kommunikationskanal, der vom Designer eines Systems gar nicht zur Kommunikation gedacht worden ist, sondern etwa zur Steuerung. Auch können verdeckte Kanäle in den Kontext der Multilevel Security (MLS) gestellt werden. Bei Multilevel Security-Systemen existieren Regeln, die bestimmte Kommunikationen verbieten, so verhindert das Bell-LaPadula-Modell beispielsweise, dass ein Prozess eines hohen Sicherheitslevels (High-Prozess) vertrauliche Daten an einen Prozess eines geringen Sicherheitslevels (Low-Prozess) schreibt bzw. dass der Low-Prozess Daten vom High-Prozess lesen kann (Anderson 2008; Bishop 2002).

Heute betrachtet man verdeckte Kanäle vor allem im Kontext der unüberwacht stattfindenden Kommunikation und verwendet die Techniken etwa für die Steuerung von Botnetzen (Li, Goyal und Chen 2008), durch deren Angriffe steigende wirtschaftliche Schäden entstehen (Fischermann, Hamann und Trauthig 2011), bei der Kommunikation von Oppositionellen oder Journalisten in überwachten Netzwerken (Zander, Armitage und Branch 2007) und gemäß Schneier wahrscheinlich auch im Bereich der Geheimdienste (Schneier 2006). Die Techniken und Bereiche, in denen verdeckte Kanäle dabei zum Einsatz kommen, variieren stark und reichen von der Unterbringung in persönlichen, elektronischen Dokumenten (Schmeh 2010) über alle erdenklichen Attribute eines Computersystems (etwa Dateinamen im Dateisystem), bis hin zu lokalen Netzen (Girling 1987) und dem Internet (Cabuk, Brodley und Shields 2004), sowie Smart-Grids (Locke und Gallagher 2010).

Neben der Steganografie, der Forschung im Bereich der Anonymität und dem Copyright Marking stellen verdeckte Kanäle einen eigenen Bereich der *Information Hiding*-Forschung dar (Petitcolas, Anderson und Kuhn 1999). Zu Grunde legen lässt sich dem Information Hiding ein von Simmons (1984) eingeführtes Modell, das sog. *Prisoner's Problem*.[1] Grundlage für das Prisoner's Problem ist die Situation zweier Gefangener, *Alice* und *Bob*, die in getrennten Zellen eines Gefängnisses untergebracht sind. Beiden Häftlingen wird ein Informationsaustausch ermöglicht, wobei die Nachrichten zwischen Alice und Bob von einem Wärter *Walter* überbracht werden. Walter kann im einfachsten Fall den gesamten Nachrichtenaustausch zwischen Alice und Bob lesen (passiver Wärter, *passive warden*), kann diese Nachrichten unter Umständen aber auch manipulieren (aktiver Wärter, *active warden*) oder fälschen (*malicious warden*) (Craver 1998). Ziel von Alice und Bob ist es nun, mit Hilfe von Information Hiding-Techniken eine von Walter unbemerkte Kommunikation zu etablieren. Selbstverständlich bietet die Steganografie unzählige Methoden, dieses Ziel zu erreichen, was sich aus historischen Berichten wie auch aus aktuellen Fachbüchern erschließt. Zusätzlich zum üblichen Fokus der Steganografie lässt sich das Prisoner's Problem allerdings hervorragend auf verdeckte Kanäle übertragen. Seien Alice und Bob beispielsweise zwei Rechner in einem überwachten Netzwerk, und möchten sie unbemerkt miteinander kommunizieren, können sie legitim und unauffällig wirkende Daten zwischen sich übertragen. In diesen Daten können bestimmte, *nicht zur Kommunikation gedachte* Attribute modifiziert werden, die die eigentlichen geheimen Informationen repräsentieren.

1 Das Prisoner's Problem ist nicht mit dem ähnlich benannten Prisoner's Dilemma aus der Spieltheorie zu verwechseln.

Bei der Repräsentation der geheimen Daten eines verdeckten Kanals wird generell zwischen zeitlichen Daten und Speicherdaten unterschieden. So können Alice und Bob etwa durch die künstliche Manipulation von Datenreihenfolgen (Netzwerkpakete, die absichtlich in der falschen Reihenfolge abgesendet wurden) oder Zeitabständen (etwa die Zeit, die einer der Teilnehmer wartet, bis ein neues Netzwerkpaket verschickt wird) miteinander kommunizieren. Man spricht bei diesen zeitlichen Manipulationen von verdeckten Zeitkanälen (*covert timing channels*). Im Gegensatz zu den Zeitkanälen basieren Speicherkanäle (*covert storage channels*) darauf, momentan nicht benötigte Attribute zu verwenden. Typische Beispiele hierfür sind etwa die Manipulation der lokalen CPU-Auslastung, das Vorhandensein von bestimmten Dateinamen in Verzeichnissen, sowie eine kaum zählbare Menge an Attributen verschiedener TCP/IP-Netzwerkprotokolle (etwa IP ID in IPv4).

Verdeckte Kanäle können nicht nur lokal, sondern auch im Netzwerk auftreten. Darüber hinaus gibt es weitere Unterscheidungen verdeckter Kanäle. Ein verdeckter Kanal ist rauschfrei, wenn seine Kommunikation ungestört verläuft, dass heißt, es gehen weder Informationen im Kanal verloren, noch kommen neue Informationen während der Datenübertragung hinzu. Die Daten kommen genau so beim Empfänger an, wie der Sender sie verschickt. Rauschende Kanäle sind in der Praxis allerdings häufig, da viele von verdeckten Kanälen verwendete Attribute durch Dritte überschrieben werden können (etwa die Time-to-live bei IPv4, die durch Hops auf dem Pfad vom Sender zum Empfänger modifiziert werden kann).

Verdeckte Kanäle können zudem in passive und aktive Kanäle unterteilt werden. Während aktive Kanäle selbst den zu übertragenden Traffic erzeugen, wird dieser Traffic bei passiven Kanälen durch Dritte erzeugt und für die verdeckte Kommunikation modifiziert. Rutkowska (2004) stellte ein solches Verfahren vor, bei dem die *Initial Sequence Number* (ISN) von TCP-Verbindungen so modifiziert wurde, dass in ihr versteckte Daten untergebracht werden konnten. Ausgenutzt wurden dabei vom Benutzer neu erstellte Verbindungen (etwa für den Besuch auf Webseiten). Der Nachteil eines solchen Verfahrens besteht darin, von anderweitig erzeugten Daten abhängig zu sein und nicht Just-in-Time senden zu können.

Techniken der verdeckten Kanäle

Für verdeckte Kanäle existieren viele verschiedene Techniken, sowohl für lokale, aber besonders für netzwerkweite Datenübertragung. Die Unterbringung versteckter Informationen in Netzwerkprotokollen ist dabei besonders stark untersucht wurden. Entsprechend gibt es für fast jedes Netzwerkprotokoll Veröffentlichungen, die die Verwendung bestimmter Headerbereiche solcher Protokolle hinsichtlich ihrer Tauglichkeit für verdeckte Kanäle untersuchen. Für die Netzwerkprotokolle, für die solche Untersuchungen noch nicht existieren, ist es nicht schwer, bisherige Verfahren zu adaptieren. Diese einfache Adaption basiert darauf, dass das Grundprinzip (versteckte Daten in derzeit nicht verwendeten Headerbereichen unterbringen) überall dasselbe ist. Im Folgenden möchten wir nur zwei Beispieltechniken für verdeckte Kanäle erläutern, wobei sich beide auf die Netzwerkübertragung konzentrieren. Der erste verdeckte Kanal verwendet dabei, ungenutzte Headerbereiche und ist somit ein *covert storage channel*. Der andere verdeckte Kanal manipuliert zeitliches Verhalten und ist somit ein *covert timing channel*.

HTTP User-Agent

Das Hypertext Transfer Protocol (HTTP) bildet die Grundlage der Datenübertragung im Web. Webbrowser kommunizieren über HTTP mit Webservern und übertragen dabei Klartext-Header wie den folgenden:

```
GET /index.html HTTP/1.1
Host: www.myserver.xyz
User-Agent: Mozilla/5.0
 (X11; Linux x86_64; rv:7.0.1)
 Gecko/20100101 Firefox/7.0.1
```

Dieser einfache HTTP Request-Header enthält nur wenige der typischen Bestandteile, ist aber für die Erläuterung eines verdeckten Speicherkanals ausreichend. Der Request besteht in diesem Fall aus einer Anfrage des Clients mit der GET-Methode (das ist eine einfache Datenabfrage). Der Client fordert dabei die Ressource »/index.html« auf dem Host »www.myserver.xyz« an (auf einem Server können mehrere Hosts parallel konfiguriert sein, sogenannte »virtual Hosts«) und verwendet dabei die HTTP-Version 1.1. Der Client gibt an, dass es sich bei ihm um einen Firefox-Browser der Version 7.0.1 handelt, wobei noch zusätzliche Software-Details (etwa, dass ein 64-Bit Linux-System verwendet wurde) mitgesendet werden.

Die Unterbringung versteckter Daten ist im Falle solcher Plaintext-Headerbereiche mit variabler Länge und vielen möglichen und zugleich legitim erscheinenden Inhalten sehr einfach. HTTP bietet neben dem User-Agent viele weitere mögliche Header-Bereiche, in denen sich Daten verstecken lassen, doch konzentrieren wir uns hier nur auf diesen einen, da das Prinzip überall gleich ist (dies gilt nicht nur für HTTP, sondern beispielsweise auch für SMTP und NNTP). Weitere Details zu Covert Channels auf der Basis des User-Agent und anderen Feldern in HTTP finden sich beispielsweise in Dyatlov und Castro (2003) und Smeets und Koot (2006).

Der einfachste Fall verdeckter Datenübertragung würde beim HTTP User-Agent darin bestehen, zwei verschiedene Browser im Feld »User-Agent« einzutragen. Wird der Internet-Explorer verwendet, steht dies für ein Bit mit dem Wert 0, der Firefox steht hingegen für ein Bit mit dem Wert 1. Wie Ihnen sicherlich auffallen dürfte, ist dieser einfache Kanal in mehrfacher Hinsicht suboptimal:

1. Fragt ein Client mit derselben IP ständig mit zwei verschiedenen Browser-Versionen an und handelt es sich bei der angefragten Ressource womöglich auch ständig um dieselbe Ressource (also »/index.html«), ist dies als Auffälligkeit zu werten und könnte den verdeckten Kanal verraten.
2. Die Unterscheidung zwischen Internet Explorer und Firefox ermöglicht nur die Übertragung von einem Bit pro Request, was nicht nur eine sehr kleine Datenmenge ist, sondern gegenüber dem Overhead des restlichen HTTP-Headers (plus sonstiger Header, wie dem TCP-Header) verschwindend gering ausfällt.

Diese beiden Probleme lassen sich dadurch angehen, dass nicht immer dieselbe Ressource angefragt wird, unterschiedliche Quell-IPs verwendet werden und gewisse zeitliche Abstände zwischen den Requests liegen. Die Übertragung größerer Datenmengen als einem Bit pro Request lässt sich beispielsweise dadurch lösen, dass vier statt zwei Browser verwendet werden können (etwa Internet Explorer, Firefox, Opera und Chrome). Somit können pro Request bereits zwei statt einem Bit übertragen werden. Wesentlich vorteilhafter ist hingegen die Manipulation mehrerer paralle-

ler Werte, etwa Betriebssystem, Architektur, Werte bezüglich der Kompatibilität zu anderen Browsern und Browser-Version. Allein durch die Vielzahl parallel existierender Firefox-Versionen ergeben sich viele Bits pro Request. Zusätzlich können durch die Anzahl an Whitespaces zwischen einzelnen Zeichen und die Unterscheidung zwischen Groß- und Kleinschreibung verdeckte Kanäle realisiert werden (Zander, Armitage und Branch 2007).

Empfangsseitig müssen beim Webserver entweder die Logdateien ausgewertet und die darin enthaltenen User-Agent-Werte gelesen werden, oder es muss ein serverseitiges Skript den Request-Header verarbeiten. Auch ist es möglich, durch einen Man-in-the-Middle-Empfänger (etwa einen Router, der zwischen Sender und Empfänger platziert ist), die User-Agents zu lesen. Solche MITM-Szenarien gibt es im Übrigen nicht nur für den Empfänger, sondern auch für den Sender, wie im Fall der passiven verdeckten Kanäle in Abschnitt erläutert.

Zeitintervalle von Netzwerkpaketen

Neben den vielen Möglichkeiten für Covert Storage Channels bieten Paketreihenfolgen und Intervallzeiten zwischen einzelnen Netzwerkpaketen ebenfalls Möglichkeiten zur versteckten Informationsübertragung. Wir möchten an dieser Stelle ein grundlegendes Verfahren vorstellen, wie es von Cabuk, Brodley und Shields (2004) eingeführt wird.

Die Autoren schlagen dabei vor, den Intervallzeiten zwischen Paketen Bit-Werte zuzuweisen. Dauert das Intervall zwischen zwei Paketen etwa eine Sekunde, ist dies ein Bit mit dem Wert 0, dauert es eine halbe Sekunde, repräsentiert es ein Bit mit dem Wert 1. Feinere Abstufungen ermöglichen aus zweierlei Gründen höhere Datenübertragungen: Zum einen werden mehr Pakete pro Sekunde übertragen, zum anderen können mehrere Bit-Werte mehreren möglichen Intervallwerten zugewiesen werden (etwa 0,1 Sekunden = 000, 0,2 Sekunden = 001, 0,3 Sekunden = 010 usw.). Durch Übertragungsfehler und Verzögerungen im Netz, also durch Rauschen, können feiner granulare Werte allerdings auch leichter verfälscht werden, womit der verdeckte Kanal falsche Werte auf Empfangsseite liefern kann. Abhilfe können in diesem Fall Methoden zur Fehlererkennung und -korrektur bei Datenübertragungen, zum Beispiel Parity-Bits schaffen. Auch sind Re-Synchronisierungen durch Preambeln (vordefinierte Intervallfolgen) ein bekanntes Mittel für die Sicherstellung qualitativer verdeckter Zeitkanäle. Zudem sind bestimmte Startzeiten für neue Übertragungsintervalle ein Mittel, mit de-synchronisierten Kanälen umzugehen, setzen allerdings möglichst genaue Uhrzeiten bei Sender und Empfänger voraus. Weitere Synchronisierungsmethoden finden sich in Cabuk, Brodley und Shields (2004).

Maßnahmen gegen verdeckte Kanäle

Bei den Maßnahmen, die gegen verdeckte Kanäle zum Einsatz kommen, werden drei Kategorien unterschieden: Eliminierung, Limitierung und Detektion. Auch in diesem Fall möchten wir uns aus Platzgründen auf einige verbreitete Methoden konzentrieren.

Erste Ansätze für Gegenmaßnahmen stammen bereits aus den 1980er Jahren. Hierzu zählt insbesondere die *Shared Resource Matrix* in Kemmerer (1983). Die Shared Resource Matrix (SRM) lässt sich auf verschiedene Bereiche des Software Development Lifecycle (SDL) anwen-

den, so kann sie sowohl auf Requirements, die in natürlicher Sprache vorliegen, als auch auf Quellcode angewandt werden.

Bei diesem Verfahren wird überprüft, unter welchen Bedingungen eine Prozedur auf eine *Ressource*, etwa eine Variable, zugreifen kann. Anschließend wird überprüft, ob der Zugriff der Prozedur lesend und/oder modifizierend erfolgen kann. Mithilfe der gewonnenen Informationen wird eine Matrix erstellt, deren Spalten die Operationen bilden und deren Zeilen die Attribute bilden (vgl. Wendzel 2012 (im Druck)). In die entsprechenden Zellen wird eingetragen, wie der Zugriff erfolgt (Bishop 2002). Kann ein High-Prozess ein Attribut mit einer Prozedur modifizieren und ein Low-Prozess dasselbe Attribut über eine Prozedur lesen, existiert ein verdeckter Kanal. McHugh (1995) erweiterte das Konzept zur *extended* Shared Resource Matrix indem er beispielsweise unabhängige Informationsflüsse separat abbildete und so vermeintliche verdeckte Kanäle als nicht existent herauskristallisieren konnte (Wendzel 2012 (im Druck)).

Ebenfalls Kemmerer war es, der zusammen mit Porras 1991 die *Covert Flow Trees* entwickelte (Kemmerer und Porras 1991). Im Gegensatz zur SRM lassen sich Covert Flow Trees (CFT) nicht auf mehrere Bereiche des SDL anwenden, sind aber dafür für die Entdeckung verdeckter Speicherkanäle in Quellcode konzipiert und erkennen in diesem Zusammenhang beispielsweise auch indirekte verdeckte Kanäle. Ein indirekter Kanal wäre etwa, wenn ein Low-Prozess eine Prozedur aufrufen muss, die erst eine andere Prozedur aufrufen muss, um einen Informationskanal herzustellen. Covert Flow Trees bilden gemäß ihrem Namen tatsächlich eine baumartige, grafisch repräsentierbare Struktur, die die Informationsflüsse in einem Programm visualisiert. Das letztliche Ergebnis einer CFT-Analyse sind allerdings Informationsfluss-Listen, die zeigen, welche möglichen verdeckten Kommunikationswege vom Sender zum Empfänger möglich sind.

Neben diesen beiden *Eliminierungs*techniken, der Shared Resource Matrix und den Covert Flow Trees, wurden auch bekannte Verfahren entwickelt, die die Kapazität eines verdeckten Kanals *limitieren* sollen. Die wichtigste Entwicklung dieser Art ist die vom Naval Research Laboratory entwickelte Pump (Naval Research Laboratory 2009).

Die *Pump* ist ein System, das auf dem Kommunikationsweg eines High-Prozesses mit einem Low-Prozess installiert wird, sie wurde in Kang und Moskowitz (1993) vorgestellt. Die Arbeitsweise der Pump ist relativ simpel: Ein Low-Prozess kann Daten an einen High-Prozess senden, wobei die Pump diese Daten zwischenspeichert und an den High-Prozess weiterleitet. Der High-Prozess darf keine Nutzdaten an den Low-Prozess schicken (die Pump erzwingt eine Einwegkommunikation mit Einschränkung). Stattdessen darf der High-Prozess Bestätigungsnachrichten (ACKs) an den Low-Prozess versenden, um zu signalisieren, dass Daten vom Low-Prozess angekommen sind. Die Pump entscheidet über Verzögerungen der ACK-Nachrichten an den Low-Prozess, sodass der High-Prozess nicht uneingeschränkt über die Datenrate der ACK-Nachrichten an den Low-Prozess bestimmen kann. Dadurch wird ein verdeckter Zeitkanal zwar nicht verhindert, doch wird seine Kanalkapazität reduziert. Es gibt verschiedene Arten der Pump, etwa die *linear quantized pump* und die *logarithmic quantized pump*, die sich primär dadurch von der normalen Pump unterscheiden, wie sie Erhöhungen und Drosselungen der Datenraten

umsetzen (Wendzel 2012 (im Druck)).

Bevor wir diesen Abschnitt abschließen, soll noch ein Vertreter für eine *Detektions*methode verdeckter Kanäle vorgestellt werden. In diesem Fall handelt es sich um eines der vielen Verfahren zur Zeitkanal-Detektion – tatsächlich gibt es mehr Detektionsverfahren für verdeckte Zeitkanäle als für verdeckte Speicherkanäle. Die Verfahren in diesem Bereich unterscheiden sich stark, so arbeiten Zander, Armitage und Branch (2007) etwa mit Klassifikationsalgorithmen und Cabuk, Brodley und Shields (2004) mit statistischen Methoden. Im Folgenden wird ein relativ einfaches Verfahren von Berk, Giani und Cybenko (2005) beschrieben:

Die Idee hinter dem Verfahren ist einfach: Ein Monitoring-System zeichnet Paketintervalle im Netzwerk auf. Dadurch kann das Monitoring-System herausfinden, wie die typischen Intervallzeiten im jeweiligen Netzwerk verteilt sind. Gemäß Berk, Giani und Cybenko werden sich bei normalem Traffic die meisten Werte um einen bestimmten Bereich (Median) häufen. Bei verdeckten Zeitkanälen, die beispielsweise zwei Zustände (Bit 0 bei 0,1 Sek. Verzögerung und Bit 1 bei 0,9 Sek. Verzögerung) kennen, ist dieser Fall hingegen nicht gegeben. Stattdessen werden (um beim genannten Beispiel zu bleiben) besonders viele Pakete bei den Intervallzeiten um den Bereich 0,1 Sekunden und 0,9 Sekunden auftreten (Wendzel 2012 (im Druck)). Zur Bewertung eines zu untersuchenden Paketverkehrs berechnen die Autoren die Kenngröße $P_{CovChan}$:

$$P_{CovChan} = 1 - \frac{C}{C_{max}} \qquad (1)$$

Hierbei ist C die Paket-Anzahl beim Durchschnittswert der Paket-Intervallzeiten und C_{max} ist die höchste Paketanzahl der gesamten Messreihe. Bei normalem Verkehr (d.h. alle Intervallzeiten sammeln sich um den zentralen Punkt) hat C in etwa denselben Wert wie C_{max}, so dass $P_{CovChan}$ nahe Null liegt. Durch die separaten Auftrittshäufungen bei verdeckten Zeitkanälen (die, wie erläutert, mindestens zwei Bereiche haben in denen besonders viele Paketintervallzeiten liegen) wird C_{max} hingegen nicht mit C übereinstimmen.

Fortgeschrittene Techniken

Bereits in den späten neunziger Jahren wurden Ansätze entwickelt, um verdeckte Kanäle komplexer zu gestalten und schwerer detektierbar zu machen. Daemon9 (1997) beschreibt den ersten Proof-of-Concept Code für einen covert channel, der auf Befehl hin das verwendete Netzwerkprotokoll wechseln konnte (entweder UDP oder ICMP). Dieser Ansatz wurde zehn Jahre später zum Konzept der *Protocol Hopping Covert Channels* erweitert (Wendzel 2008), die den Protokollwechsel randomisiert und transparent durchführen konnten, sowie durch ein internes *Mikroprotokoll* für die Sortierung des Traffics auf Empfangsseite sorgen konnten.

Ray und Mishra (2008) präsentierten ein Jahr später ein erstes speziell für den Einsatz von verdeckten Kanälen entwickeltes Mikroprotokoll. Zuvor gab es zwar bereits ein Verfahren von Stødle (2009), der ein ähnliches Protokoll in Ping Tunnel implementierte, doch war dieses nicht so platzsparend, wie das Protokoll von Ray und Mishra.

Yarochkin, Shih-Yao, Chih-Hung u. a. (2008) konnten ein Verfahren vorstellen, mit dem es verdeckten Kanälen möglich wurde, ihre Kommunikationsprotokolle zu detektieren (durch passives Überwachen des Netzwerks), genannt *adaptive covert channels*.

In Wendzel und Keller (2011) erweitern wir die bisherigen Ansätze adaptiver verdeckter Kanäle und des Protocol Hoppings, indem wir eine intelligente und optimierte Auswahl von Netzwerkprotokollen (abhängig von dem Platz, der pro Protokoll zur Verfügung steht) einführen und verdeckte Kanäle in einem Overlay Netzwerk operieren lassen. So lässt sich etwa eine Optimierung dafür erzielen, eine Datenübertragung mit möglichst wenigen Netzwerkpaketen bzw. mit möglichst wenig Overhead durchzuführen. Weiterhin ist ein intelligenter Austausch von Informationen unterstützter Netzwerkprotokolle zwischen Peers im Overlay-Netzwerk vorgestellt worden. Zu diesem Zweck werden Mikroprotokolle verwendet, über die Metainformationen (Version der Covert Channel Software, unterstützte Protokolle etc.) ausgetauscht werden, womit *Backward-Compatibility* ermöglicht wird. Die Infrastruktur verdeckter Overlay-Netzwerke ist somit schrittweise aktualisierbar.

Zusammenfassung

Verdeckte Kanäle sind ein *Dual-Use*-Gut und können sowohl für Angriffe (etwa zur Koordination von Botnetzen oder im Bereich Data Leakage bei Unternehmen) verwendet werden wie für freie Meinungsäußerungen (etwa durch Journalisten in überwachten Netzen). Über die tatsächliche Verwendung der verdeckten Kanäle ist (außerhalb von Botnetzen) wenig bekannt, da Anwender dieser Techniken (wie auch im Bereich Steganografie) nicht darüber kommunizieren dürfen — schließlich geht es um das Verbergen der Kommunikation an sich, nicht nur um das Verbergen von Inhalten.

Die Vielfältigkeit der Techniken für verdeckte Kanäle ist besonders im Bereich der TCP/IP-Protokolle groß. Die Detektion (besonders von storage channels) ist schwierig, für verdeckte Zeitkanäle existieren hingegen viele Ansätze. Präventions- und Limitierungsansätze sind ebenfalls vorhanden, aber – etwa mit Ausnahme der Pump – meist auf sehr spezielle Problemfälle ausgelegt. Die Detektierbarkeit verdeckter Kanäle steigt mit deren Bandbreite und generell werden nicht überwachte/verhinderte verdeckte Kanäle mit maximal 1 Bit/Sek. auch in Hochsicherheitssystemen toleriert. Neue Techniken wie der optimierte Wechsel des von verdeckten Kanälen verwendeten Protokolls werden die Detektion in Zukunft erschweren.

Über die Autoren

Steffen Wendzel studierte Informatik an den Hochschulen Kempten und Augsburg. Er ist externer Doktorand am Lehrstuhl für VLSI und Parallelität der FernUniversität in Hagen und wissenschaftlicher Mitarbeiter der Hochschule Augsburg, sowie Autor mehrerer Fachbücher aus den Bereichen Linux und IT-Sicherheit. Webseite: *http://www.wendzel.de*.

Jörg Keller studierte Informatik und Mathematik an der Universität des Saarlandes in Saarbrücken, promovierte dort mit einem Thema aus der Parallelverarbeitung und habilitierte sich dort nach einem Postdoktorandenjahr am CWI (niederl. Forschungszentrum für Mathematik und Informatik) in Amsterdam. Seit 1996 leitet er als Professor der FernUniversität in Hagen das Lehrgebiet Parallelität und VLSI. Seine Forschungsinteressen umfassen IT-Sicherheit und Parallelverarbeitung sowie Fehlertoleranz und internetgestützte Fernlehre.
Webseite: *www.fernuni-hagen.de/pv*

Literaturverzeichnis

Anderson, R. (2008). *Security Engineering: A guide to build dependable distributed systems*. Wiley. Zugriff am unter `http://www.cl.cam.ac.uk/~rja14/book.html`

Berk, V., Giani, A. & Cybenko, G. (2005). *Detection of Covert Channel Encoding in Network Packet Delays* [Technical report tr536].

Bishop, M. (2002). *Computer Security: Art and Science*. Readin: Addison Wesley.

Cabuk, S., Brodley, C. E. & Shields, C. (2004). IP Covert Timing Channels: Design and Detection. In *Proc. 11th ACM Conference on Computer and Communications Security (CCS '04)* (Seiten 178–187).

Craver, S. (1998). On Public-Key Steganography in the Presence of an Active Warden. In *Proc. Information Hiding* (Band 1525, Seiten 355–368). Berlin: Springer.

Daemon9. (1997). LOKI2: the implementation. *Phrack Magazine*, *7*.

Dyatlov, A. & Castro, S. (2003). Exploitation of data streams authorized by a network access control system for arbitrary data transfers: Tunneling and covert channels over the HTTP protocol, v1.0. Zugriff am unter `http://dl.packetstormsecurity.net/papers/protocols/covert_paper.txt`

Fischermann, T, Hamann, G & Trauthig, J. (2011). Erpresser im Netz. *Die Zeit*, *38*, 27.

Girling, C. G. (1987). Covert Channels in LAN's. *IEEE Transactions on Software Engineering*, *SE-13*, 292–296.

Kang, M. H. & Moskowitz, I. S. (1993). A Pump for Rapid, Reliable, Secure Communication. In *Proceedings of the 1st ACM Conference on Computer and Communication Security* (Seiten 119–129).

Kemmerer, R. A. (1983). Shared resource matrix methodology: An approach to identifying storage and timing channels. *ACM Transactions on Computer Systems*, *1*, 256–277.

Kemmerer, R. A. & Porras, P. A. (1991). Covert Flow Trees: A Visual Approach to Analyzing Covert Storage Channels. *IEEE Transactions on Software Engineering*, *17*, 1166–1185.

Lampson, B. (1973). A Note on the Confinement Problem. *Communications of the ACM*, *16*, 613–615.

Li, Z., Goyal, A. & Chen, Y. (2008). Honeynet-based Botnet Scan Traffic Analysis: Advances in Information Security/Botnet Detection. In *Proc. Advances in Information Security* (Band 36, Seiten 25–44). Springer.

Locke, G. & Gallagher, P. D. (2010). NISTIR 7628 – Guidelines for Smart Grid Cyber Security.

McHugh, J. (1995). *Covert Channel Analysis*.

Naval Research Laboratory. (2009). Network Pump Brochure.

Petitcolas, F. A. P, Anderson, R. J. & Kuhn, M. G. (1999). Information hiding: A survey. In *Proceedings of the IEEE* (Band 88, Seiten 1062–1078).

Ray, B. & Mishra, S. (2008). A Protocol for Building Secure and Reliable Covert Channel. In *Proc. Sixth Annual Conference on Privacy, Security and Trust (PST 2008)* (Seiten 246–253).

Rutkowska, J. (2004). The implementation of passive covert channels in the Linux kernel. In *21st Chaos Communication Congress*. Berlin.

Schmeh, K. (2010). Covert Channels in elektronischen Ausweisen. In C. Paulsen (Herausgeber), *Sicherheit*

in vernetzten Systemen: 17. DFN Workshop (Seiten B1–B8).

Schneier, B. (2006). *Angewandte Kryptographie: Protokolle, Algorithmen und Sourcecode in C* (2. Aufl.).

Simmons, G. J. (1984). The Prisoner's Problem and the Subliminal Channel. In *Advances in Cryptology: Proceedings of CRYPTO '83* (Seiten 51–67). Plenum Press.

Smeets, M. & Koot, M. (2006). *Covert Channels: Research Report for RP1*.

Stødle, D. (2009). Ping Tunnel: For those times when everything else is blocked. Zugriff am unter `http://www.cs.uit.no/~daniels/PingTunnel/`

Wendzel, S. (2008). Protocol Hopping Covert Channels. *Hakin9*, 20–21.

Wendzel, S. (2012 (im Druck)). *Tunnel und verdeckte Kanäle im Netz: Grundlagen, Protokolle, Sicherheit und Methoden*. Vieweg+Teubner.

Wendzel, S. & Keller, J. (2011). Low-attention forwarding for mobile network covert channels. In *Proc. IFIP Communications and Multimedia Security (CMS 2011)* (Seiten 122–133).

Yarochkin, F. V., Shih-Yao, D., Chih-Hung, L. (2008). Towards Adaptive Covert Communication System. In *Proc. 14th IEEE Pacific Rim International Symposium on Dependable Computing* (Seiten 153–159).

Zander, S., Armitage, G. & Branch, P. (2007). Covert Channels and Countermeasures in Computer Networks. *IEEE Communications Magazine, 45*, 136–142.