STEFFEN WENDZEL

Protocol Hopping Covert Channels

Schwierigkeitsgrad:

Die Detektion von verdeckten Kommunikationskanälen kann durch Protokollwechsel erschwert werden. Dieser Artikel zeigt, wie es geht.

IN DIESEM ARTIKEL ERFAHREN SIE…

was Protocol Hopping Covert Channel sind

wie diese realisiert werden können und was es dabei zu beachten gibt

was das Besondere an Protocol Hopping Covert Channeln ist

WAS SIE VORHER WISSEN/KÖNNEN SOLLTEN…

TCP/IP Grundkenntnisse;

Tunneling-Grundkenntnisse;

Grundkenntnisse über IDS.

The Prisoners Problem

Als Covert Channel bezeichnet man – vereinfacht ausgedrückt – einen verdeckten Kommunikationskanal. Dieser Kommunikationskanal läuft über ein Kommunikationsmedium, welches dritten zugänglich ist (etwa Schall). Dies wird von G. J. Simmons im so genannten *Prisoners Problem* verdeutlicht (siehe [S84A]). Dabei geht es kurz gesagt darum, dass zwei Gefangene ihren Ausbruch planen. Beide Gefangene werden allerdings in isolierten Zellen eingesperrt und jede Kommunikation muss über den Wärter geleitet werden, der die zu tauschenden Informationen lesen, fälschen und vor dem anderen Gefangenen geheimhalten kann.

Um dennoch ihren Ausbruch zu planen, müssen die Gefangenen innerhalb der eigentlich zu übertragenen Informationen (etwa in der Schrift auf einem Stück Papier) versteckte Informationen einbauen, mit denen sie den Fluchtplan besprechen können. Jedem Informatiker würden sicherlich gleich mehrere Techniken einfallen, dies zu realisieren (etwa könnte man die vier Ecken eines Papierstücks unauffällig einknicken und als Bitmuster verwenden um Buchstaben des Alphabets zu übertragen). Damit wurde dann auch schon das Prinzip eines Covert Channels angewandt: Informationen innerhalb von anderen Informationen möglichst erfolgreich und unentdeckt zu übertragen.

In der 2007er Januar-Ausgabe der Hakin9 (siehe [W07B]) habe ich bereits ausführlich über Tunneling-Techniken gesprochen, und das es leicht möglich ist, Daten in verschiedensten Protokollen zu verstecken und durch diese zu tunneln, daher werde ich nun nicht noch einmal auf dieses spezielle, aber hiermit zusammenhängende Thema eingehen (weitere Informationen finden sich auch in [PW07A]).

Neben den darin beschriebenen Techniken, Daten zu tunneln, existieren noch weitere Möglichkeiten, Covert Channels zu implementieren (etwa die vielfach beschriebenen Timing-Channels oder auch die von Joanna Rutkowska beschriebenen Passive Covert Channels, die keinen eigenen Traffic erzeugen, sondern mit einer modifizierten TCP Initial Sequence Number arbeiten). Mehr zu diesen Techniken finden Sie in den Literaturverweisen am Ende des Artikels.

Im November 2007 habe ich eine Technik beschrieben, die ebenfalls einen etwas anderen Weg einschlägt: Statt Daten durch *ein* Protokoll zu tunneln, werden *mehrere* Protokolle parallel verwendet.

Protocol Hopping Covert Chanels im Detail

Der Wechsel des zu Grunde liegenden Übertragungsprotokolls wird dabei eingesetzt, um die Detektion seitens eines Intrusion Detection Systems zu erschweren. Ausserdem soll dadurch die Rekonstruktion des Covert Channels erschwert werden.

Möchte man verschiedene Protokolle zur Übertragung eines Covert Channels nutzen, dann gibt

es allerdings einige Probleme, die es zunächst zu eliminieren gilt. Im Folgenden werde ich diese Probleme aufzeigen.

Da unterschiedliche Protokolle unterschiedliche Möglichkeiten zum Verstekken der Daten des Kommunikationskanals bieten, kann ein *Protocol Hopping Covert Channel* immer nur so sicher sein, wie sein schwächstes Glied (also das Protokoll mit den schlechtesten Versteckmöglichkeiten). Das ist ein klarer Schwachpunkt der *Protocol Hopping Covert Channel* und muss daher bei einer Implementierung intensiv bedacht werden.

Damit kommt man auch geradewegs automatisch zur zweiten Problematik: Unterschiedliche Protokolle bieten nicht nur qualitativ unterschiedliche Versteckmöglichkeiten in Bezug auf die Detektion, sondern auch unterschiedlich viel Platz! Es gibt Protokolle bei denen man relativ problemfrei mehrere Bytes pro Paket tunneln kann und es gibt Protokolle, die nur wenigen Bits Platz bieten. Protokolle mit Platzproblemen können immer dann nur unter großem Risiko zum Einsatz kommen, wenn große Datenmengen übertragen werden sollen. Denn desto mehr Daten versteckt übertragen werden, desto wahrscheinlicher ist, dass es jemandem auffällt.

Mikro-Protokoll

Des Weiteren unterstützen nicht alle Protokolle Reliability-Mechanismen, wie etwa TCP. Würde man nur TCP-basierte Protokolle (etwa HTTP oder SMTP) benutzen, um einen *Protocol Hopping Covert Channel* zu implementieren, so hätte sich das Reliability-Problem gelößt und TCP würde sich darum kümmern, ob Pakete korrekt und in der – aus TCP-Sicht – richtigen Reihenfolge ankommen. Doch möchte man auch Protokolle ohne solchen freundlichen Service benutzen, dann kommt man um die Implementierung von eigenen Reliability-Mechanismen nicht herum.

Listing 1. *XXXXXXXXXXXXX*

```
HostA, Terminal1$ ./phcct -a IP-Of-
                      Host-B
HostB, Terminal1$ ./phcct -a IP-Of-
                      Host-A
...
HostA, Terminal2$ telnet localhost 9999
HostB, Terminal2$ telnet localhost 9999
```

Doch genau hier liegt ein Denkfehler: Die Daten kommen zwar aus der Sicht des TCP in der richtigen Reihenfolge beim Empfänger an, doch wenn mehrere parallele Verbindungen bestehen, kann auch TCP nicht wissen, in welcher Reihenfolge Pakete der verschiedenen Verbindungen gesendet wurden, es kümmert sich eben nur um die Reihenfolge der Pakete innerhalb einer Verbindung.

An dieser Stelle entwirft man Mikro-Protokolle, die eine *Protocol Hopping Covert Channel* interne ID kennen. Jedes Paket bekommt dabei eine solche ID und IDs können beispielsweise aufsteigend implementiert werden. Der Empfänger-Prozess muss dann nur noch die IDs der Pakete in die richtige Reihenfolge bringen, um sie richtig zusammensetzen zu können.

Das ganze funktioniert also praktisch umgesetzt folgendermaßen: Host A sendet über Protokoll X Datenpaket 1 (ID=1) und kurz darauf Datenpaket 2 (ID=2) über Protokoll Y. Egal, in welcher Reihenfolge die Pakete der beiden seperaten Verbindungen beim Zielhost (Host B) ankommen: Durch die interne ID im Mikroprotokoll kann der Empfänger-Prozess die Daten wieder in der richtigen Reihenfolge zusammensetzen.

Auch hier gibt es wieder ein Platzproblem, denn ein Mikroprotokoll benötigt zusätzlichen Platz, und Platz ist, wie bereits erläutert, sehr knapp bemessen.

Weitere Aufgaben des Mikroprotokolls können die Implementierung einer Checksum für den getunnelten Traffic sowie ein Feld, welches die Größe des getunnelten Payloads angibt, sein. Man muss hier allerdings bedenken, dass viele Protokolle (beispielsweise alle UDP und TCP basierten) sowieso über eine Checksum verfügen (es sei denn, man stellt im Falle von UDP kernelseitig explizit die Checksum-Berechnung aus, was bei einigen Betriebssystemen möglich ist). Ausserdem sollte nicht vergessen werden, dass man weiterhin mit einem Platzproblem zu kämpfen hat! Auch ein Feld im Mikroprotokollheader, das die Größe des geschleusten Payloads angibt, benötigt etwas Platz, den man eventuell für zusätzlichen Payload besser nutzen kann. Man könnte alternativ zu einem solchen Größenfeld im Header auch ein Abschlusszeichen hinter dem Payload einbauen oder die Größe des Payloads statisch festlegen (und dann eventuell auch ungenutzen Platz mit bestimmten Bitkombinationen auffüllen).

Sender-Reihenfolge

Wie Sie bemerkt haben werden, ist die Implementierung eines Mikroprotokolls nicht immer ganz einfach. Doch durch ein Mikroprotokoll kann ein *Protocol Hopping Covert Channel* eine seiner wichtigsten zusätzlichen Eigenschaften erreichen, die ich *Protocol Order Mixing* nenne. Dabei wird – um eine Detektion zusätzlich zu erschweren – die Reihenfolge der verwendeten Protokolle und die Reihenfolge, in der Datenpakete gesendet werden, zufällig bestimmt. Der Empfänger kann nämlich durch die ID im Mikroprotokoll-Header eine Rekonstruktion des usprünglichen Traffics bewirken. Das bedeutet wiederum, dass der Sender wirklich mit einer Zufallsfunktion das zu verwendende Protokoll aussuchen kann (der Empfänger muss schließlich nicht wissen, welches Protokoll als nächstes zu erwarten ist).

Schwerer als die zufällige Wahl des Protokolls ist allerdings das zufällige Senden der Daten zu implementieren. Oftmals müssen nur wenige Bytes oder gar Bits übertragen werden und dabei lohnt es sich schlicht oftmals nicht, diese Daten auf mehrere Pakete aufzuteilen und beispielsweise zuerst Paket 2 und dann Paket 1 zu senden. Ein intelligenter Algorithmus, der sich um diese Entscheidung kümmert, ist also von Nöten.

Literatur

- [S84A] Gustavus J. Simmons: The Prisoners' Problem and the Sublimal Channel, 1984.
- [R04A] Joanna Rutkowska: Implementation of Passive Covert Channels in the Linux Kernel, December 2004.
- [PW07A] Steffen Wendzel & Johannes Plötner: Praxisbuch Netzwerksicherheit, 2. Auflage, Galileo-Press, 2007.
- [W07A] Steffen Wendzel: *Protocol Hopping Covert Channels*. An Idea and the Implementation of a Protocol switching covert channel, November 2007, doomed-reality.org.
- [W07B] Steffen Wendzel: Firewalls umgehen mit Protokoll-Tunneling, Hakin9 01/07.

phcct

Zur Implementierung eines einfachen nichtverschlüsselten *Protocol Hopping Covert Channel*s über einige TCP-Protokolle habe ich einen kleinen Proof of Concept Code *phcct* (siehe beiliegende Hakin9-CD) implementiert. phcct unterstützt *Protocol Oder Mixing* aber kein zufällig bestimmtes Senden von Datenpaketen. Listing 1 zeigt ein Anwendungsbeispiel aus [W07A] zur Errichtung eines kleinen Chats via Telnet dessen Daten über einen *Protocol Hopping Covert Channel* übertragen werden. Nach dem Start beider Programme muss auf jedem Programm die Return-Taste gedrückt werden, damit die Verbindungen aufgebaut werden.

Passive Protocol Hopping Covert Channels

Die von Joanna Rutkowska in [R04A] beschriebenen Passive Covert Channels müssten sich, meinen Überlegungen nach, durchaus mit der Idee des Protocol Hopping zu Passive *Protocol Hopping Covert Channels* kombinieren lassen. Derzeit gibt es aber noch keine Implementierung für diese Form der verdeckten Kommunikation.

Weiterer Hinweis

Auf der CD dieser Hakin9-Ausgabe finden Sie neben dem eben beschriebenen Proof of Concept Code auch meinen zugehörigen Text *protocol hopping covert channel*s ([W07A]).