Zusammenfassung: „Unexpected Means of Protocol Inference“

Andreas Martin Floß
(andimf@cs.tu-berlin.de)

Seminar „Internet Measurement“

Technical University of Berlin, Germany

SS 2007 (Version vom 4. Juli 2007)

WS 2006/2007 (Version vom 4. Juli 2007)

1 Abstract

Mit der Kenntnis von Quell- und Zielport konnten Netzwerk-Administratoren in der Vergangenheit leicht Applikation und zugehöriges Protokoll identifizieren. Die Entkoppelung von Applikation und Ports erschwert Netzwerk-Administratoren die Kontrolle über das Netzwerk. Ma et al. [4] beschreiben ein Verfahren zum Identifizieren von bekannten Applikationen und zum Erkennen von unbekannten Applikationen im Netzwerkverkehr. Dies funktioniert ganz ohne Kenntnis von Zusatzinformationen, wie z.B. Ports oder andere Header-Informationen. Dabei bedienen sich Ma et al. drei Techniken aus dem Gebiet des Maschinellen Lernens und haben dazu ein Framework entwickelt. Obwohl ein Verfahren sehr gute Ergebnisse liefert, haben alle Verfahren Schwächen. Insgesamt zeigen aber die Ergebnisse, dass es möglich ist, eine solche Klassifizierung vorzunehmen. Den Netzwerk-Administratoren damit entscheidend geholfen werden.

2 Einleitung

Traditionell benutzen Applikationen in Netzwerken einen oder mehrere festgelegte Ports [4]. Diese Ports sind die Verbindungspunkte über die Applikationen von Host zu Host kommunizieren. Auf welche Art und Weise die Applikationen miteinander kommunizieren ist in einem Applikations-Protokoll (nachfolgend Protokoll) festgelegt.

Seit dem vermehrten Einsatz von Firewalls, dem Tunneln von Verbindungen und der Existenz neuer verteilter Applikationen, wie z.B. Skype oder Bittorrent, wird dieses Konzept der *well known ports* immer häufiger umgangen [3].

Mit der Kenntnis von Quell- und Zielport konnten Netzwerk-Administratoren leicht Applikation und das zugehörige Protokoll identifizieren. Administratoren größerer Unternehmen möchten oftmals aus Performance- , Kapazitäts- und Sicherheitsgründen die Vielfalt an Applikationen einschränken. Die Entkoppelung von Applikation und Ports erschwert Netzwerk-Administratoren die Kontrolle über das Netzwerk und der darin kommunizierenden Applikationen.

Im Paper von MA et al. werden drei Verfahren entwickelt, die eine selbstständige Identifikation von Applikationen unabhängig von Portnummern anhand von Netzwerkdaten ermöglichen. Zudem sollen unbekannte Applikationen und deren Protokolle von anderen unterschieden werden können.

Nach einer Vorstellung von Netzwerkprotokoll-Grundlagen und Definitionen folgt zunächst die Beschreibung der drei Verfahren zur Protokollerkennung. Zunächst werden zwei statistische Verfahren beschrieben: das *Product Distribution Model* und das *Markov Process Model*. Danach folgt das strukturelle Modell des *Common Substring Graph* (CSG). Letztendlich werden alle drei Verfahren abschließend evaluiert und eine Zusammenfassung gegeben. Auf das ebenfalls im Paper vorgestellte Framework zur Klassifikation und auf das Erkennen unbekannter Protokolle wird in dieser Ausarbeitung nicht eingegangen.

3 Grundlagen, Definitionen und Techniken

In diesem Abschnitt werden grundsätzliche Begriffe des Netzwerkverkehrs, sowie weiterführende Definitionen und Techniken der Protokollerkennung erläutert.

3.1 Netzwerkprotokolle, Session und Flow

Die beiden wichtigsten Protokolle im Internet und in IT-Netzwerken sind *TCP* und *UDP*. Während das IP-Protokoll auf *Network-Layer*-Ebene den Weg für Pakete von Quelle zu Ziel organisiert, sind TCP und UDP für den verlässlichen und sparsamen Transport von Daten in Form von Netzwerkpaketen auf der *Transport-Layer*-Ebene verantwortlich [9].

Eine *Verbindung* zwischen zwei Hosts wird durch das TCP-Protokoll und dem darin definierten 3-Way-Handshake ermöglicht [7]. UDP wird als oftmals auch als verbindungslos bezeichnet, weil pro Verbindung nur ein Paket gesendet werden kann. Zwei Hosts können mittels einer Verbindung miteinander in beide Richtungen kommunizieren.

Diese Kommunikation von Applikationen per TCP oder UDP wird in Einheiten, den sogannten *Sessions* unterteilt. Eine Session besteht für eine bestimmte Zeitdauer. Speziell Ma et al. betrachteten nur die reinen Daten von Sessions, die Applikationen austauschen. Header- und andere Meta-Informationen werden nicht weiter betrachtet. Eine Session wird wiederum durch ein Paar von *Flows* definiert. Ein Flow-Paar besteht jeweils aus einem Byte-Strom von Daten vom *Initiator* zum *Responder* und einem Byte-Strom vom Responder zu Initiator.

Eine Session benötigt folgendes 5-Tupel an Informationen:

- Initiator-Adresse,
- Initiator-Port,
- Responder-Adresse,
- Responder-Port sowie eine
- IP-Protokollnummer.

Betrachtet man dieses 5-Tupel einer Session zusätzlich noch mit einer Richtung, dann haben wir eine Beschreibung des Flows. Somit besteht eine TCP-Session aus zwei verschiedenen Flows: einmal vom Initiator zum Responder und umgekehrt.

Beobachtet man den Netzwerkverkehr zwischen zwei Hosts und deren Ports auf Transport-Layer-Ebene, z.B. mit Wireshark[10], so kann man einzelne Sessions identifizieren. Welches Protokoll aber innerhalb einer Session zum Einsatz kommt, kann man nicht ohnes Weiteres direkt erlernen. Methoden des Maschinellen Lernens wären dafür geeignet, allerdings müssen die Protokolle untereinander verschieden sein. Um dies genauer zu betrachten, benötigt man eine für diesen Zweck spezielle Modellierung eines Protokolls.

3.2 Modellierung eines Protokolls

Ein Protokoll ist laut [4] im Modell eine Verteilung auf den ersten n Bytes von Sessions. Die daraus resultierende Verteilung hätte eine Menge von 256^{2n} möglichen diskreten Werten. Um die Komplexität weiter zu senken, wird im Modell ein Protokoll als ein Paar von Verteilungen auf den ersten n Byte der beiden Flows betrachtet, die eine Session darstellen. Es ist laut [2] völlig ausreichend die ersten 64 Bytes eines Flows anzuschauen. Daraus resultiert nun eine Flow-Länge n von 64 Bytes und eine Kombination von 256 Möglichkeiten pro Byte. Ein Protokoll, das durch zwei Verteilungen von Flows beschrieben wird, unterscheidet sich somit durch $2 \cdot 256^n$ verschiedene Werte. Diese Werte werden in einem *Feature-Vektor* zusammengefasst

3.3 Konstruieren von Protokoll und Classifier

Für eine möglichst genaue Beschreibung eines Protokolls benötigt man eine genügend große Anzahl von ähnlichen Sessions. Beim Lernen der Protokolle werden anfänglich gleichartige Sessions in Gruppen zusammengefasst. Dies ist durch zusätzliche Informationen à priory gegeben.

Aus den gleichartigen Sessions werden *Cells*. Bei der Konstruktion von Protokollen, dem Finden einer möglichst optimalen Beschreibung, werden die vorher bekannten gleichen Sessions zu einer *Cell* zusammengefasst. Ähnliche Cells werden zusammengefasst und zu einer neuen Cell zusammengefügt. Cells sind somit die Classifier für die zu vergleichenden Sessions.

Cells beschreiben ein Protokoll und werden deshalb als *Classifier* benutzt.

Um eine bisher unbekannte Session zu genau einer Cell zuzuordnen, benutzt man das *Maximum-Likelihood-Estimator-Verfahren* [5].

4 Verfahren der Protokoll-Erkennung

Ma et al. haben drei Verfahren Entwickelt um - anhand der ersten 64 Byte eines Flows - Protokolle zu identifizieren und zu klassifizieren. Dabei werden allein die Nutzdaten auf Applikationsebene betrachtet. Sämtliche Header-Informationen werden ignoriert. Pakete ohne Nutzdaten werden aussortiert. Die ersten beiden Verfahren basieren auf statistischen Eigenschaften der Protokolle und somit der Verteilungen über die ersten n Byte eines Flows.

4.1 Product Distribution Model

Im Product Distribution Model werden zwei Konzepte der Statistik genutzt: Wahrscheinlichkeitsrechnung für unabhängige Ereignisse, sowie *Relative Entropie*. Der Trick hierbei ist die relativ simple Überlegung: fixe Wahrscheinlichkeiten von unterscheidbaren Zeichen kommen an fixen Stellen in den Nutzdaten vor. Man kann vermuten, dass demnach besonders binäre Protokolle gute Ergebnisse hervorbringen.

Besteht ein Flow aus 4 Byte mit dem String „POST“, dann wäre ein anderer Flow mit „POST“ mit 100 prozentiger Wahrscheinlichkeit vom gleichen Protokoll. Wäre stattdessen nun ein 4-Byte-Flow mit „HOST“ verglichen worden, dann wäre die Wahrscheinlichkeit immer noch sehr hoch, dass das selbe Protokoll hier zugrunde liegt. Denn 3 von 4 Bytes haben eine gleiche Verteilung über die Flows. Jedes der n Bytes wird als unabhängig von den anderen betrachtet. Für jede n-te Stelle, also für alle n Bytes wird eine Verteilung über alle Flows berechnet. Multipliziert man nun alle Verteilungen erhält man eine Produktverteilung für Flows. Wegen der Annahme, dass die Verteilungen unabhängig sind, ist die Berechnung ohne großen Aufwand durch simple Multiplikation möglich. Allerdings ist dies eine sehr gewagt Annahme; sie bringt aber in der Praxis erstaunlich gute Ergebnisse. Relative Entropie ist ein Maß für die Unterscheidbarkeit zweier Verteilungen. Möchte man die Ähnlichkeit zweier Verteilungen erfahren, so muss man nur ihre Unterschiede negieren. Auf diese Weise kann man bekannte gleichartige Sessions (Gruppen) erkennen und zu Cells zusammenfügen, die einen Classifier bilden. Nun kann man berechnen, wie groß die Wahrscheinlichkeit ist, dass eine unbekannte Session zu einer Cell gehört.

4.2 Markov Process Modell

Das Markov Process Modell bedient sich ebenfalls der Wahrscheinlichkeitsrechnung mit Unabhängigkeitsannahme zwischen den verschiedenen Bytes eines Flows. Im Gegensatz zum Product Distribution Model werden nicht fixe Bytes in den Nutzdaten betrachtet. Viel mehr werden unterscheidbare Strings durch Markov-Ketten erzeugt(siehe Abbildung 1 und 2). Mittels Entropie wird ebenfalls die Unterscheidbarkeit der Strings berechnet.

Stellen wir uns einen gerichteten Graphen vor, dessen Knoten alle verschiedenen 256 Byte-Zeichen darstellt. Verbunden werden die Knoten durch Kanten, die mit einer *Änderungswahrscheinlichkeit* (transition propability) gewichtet sind. Alle Änderungswahrscheinlichkeiten $P(e_i)$ der Kanten eines Knotens v_n haben die Summe 1.

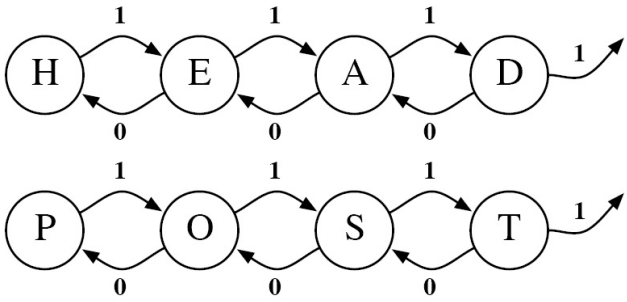

Abbildung 1: Markov Prozesse mit Bildung der Strings HEAD and POST.

Wandert man nun durch einen solchen Graphen, kann man berechnen, wie groß die Wahrscheinlichkeit ist, dass beispielsweise ein String zu einer Menge von anderen Strings gleich ist. Dieser String ist letztendlich ein Flow der Länge n. Man kann damit nun berechnen wie unterschiedlich ein Paar von Flows ist.

Diese Markov Process Modell-Methode eignet sich besonders gut für textbasierte Protokolle. Google Page Rank benutzt Markov-Prozesse um die Verlinkungshäufigkeit und somit die Bedeutung einer Website zu berechnen [1].

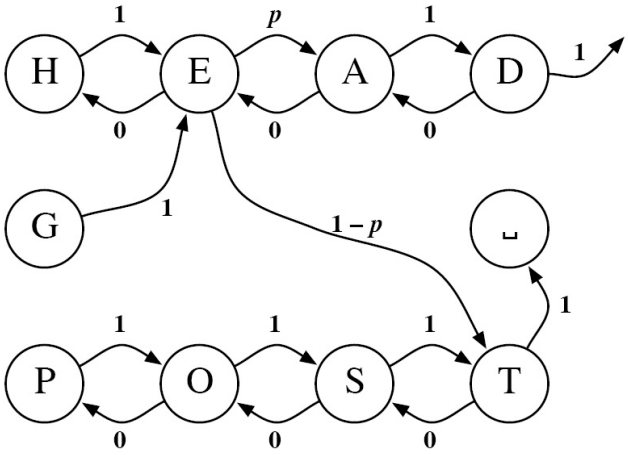

Abbildung 2: Ein neuer String GET in den Graph eingehangen.

4.3 Common Substring Graphs

Common Substring Graphs unterscheiden sich dahingehend, dass strukturelle Informationen von Flows benutzt werden. Gesucht werden nun *Common Substrings*, die häufig vorkommenden Teil-Strings in den Flows eines Protokolls (siehe Abbildung 3).

Diese Common Substrings kann man mit dem LCS-Algorithmus (Longest Common Subsequence algorithm) besonders gut berechnen, wenn man eine große Anzahl von Flows miteinander vergleicht. Zusätzlich können auch Informationen über Sequenz und Lage der der Common Substrings gewonnen werden. Die Berechnungen werden mit Hilfe des Smith-Waterman Algorithmus aus der Molekularbiologie durchgeführt [6].

5 Ergebnisse

Den Messergebnissen von [4] liegen drei verschiedene Datensätze zugrunde. Wie Abbildung 4 zeigt, gibt es eine unterschiedliche Anzahl von Flows, sowie unterschiedliche Gewichtungen der genutzten Protokolle im Netzwerkverkehr. Betrachtet man die fehlerhafte Klassifizierung (total) in Abbildung 4, dann hat das Markov Process Modell am schlechtesten abgeschnitten. Es zeigen sich fehlerhafte Klassifizierungen bis annähernd 10% aller Flows. Im

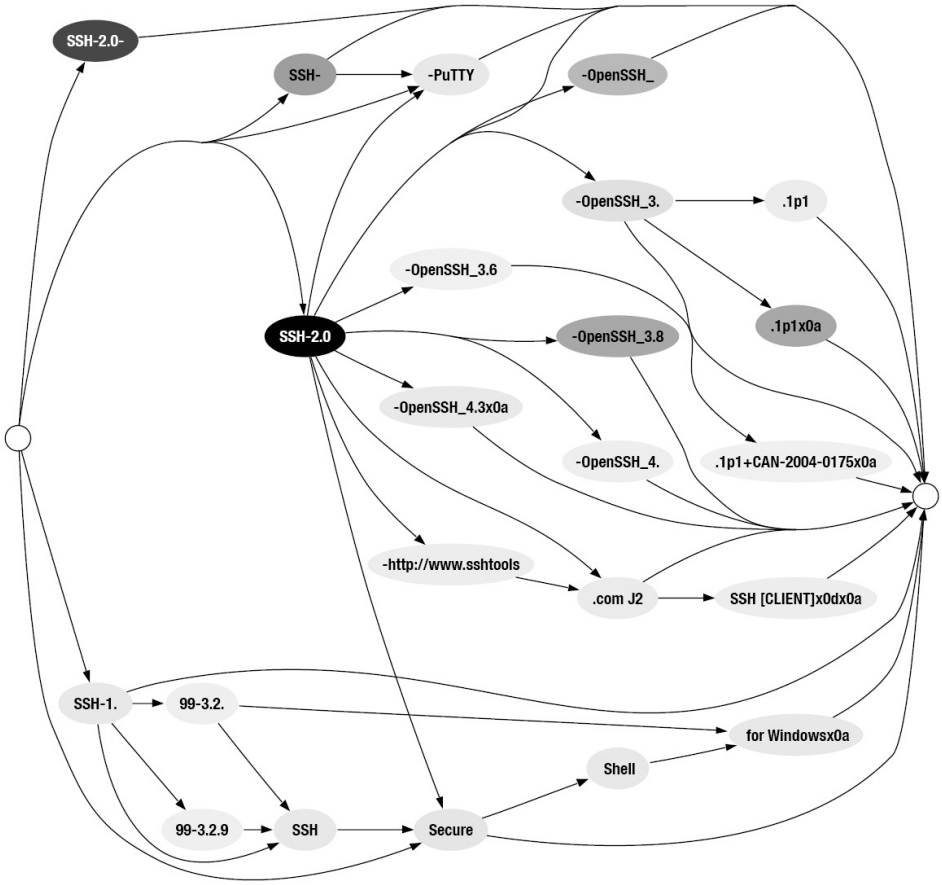

Abbildung 3: Das Modell eines Common Substring Graph für das SSH-Protokoll zeigt Teil-Strings. Je dunkler die Knoten, desto häufiger das Vorkommen dieser.

Vergleich zu den beiden anderen Verfahren kann Markov nur die Wahrscheinlichkeit von ähnlichen Strings der verschiedenen Nutzdaten vergleichen. Besonders bei binären Protokollen, wie z.B. NTB (NetBIOS) und NBNS (NetBIOS Name Service), versagt Markov laut Abbildung 5 im Vergleich zu den anderen beiden Verfahren - betrachtet man die Fehler-Rate (Err.%).

	Cambridge			Wireless			Departmental		
	total	learned	unlearned	total	learned	unlearned	total	learned	unlearned
Product	1.68%	0.50%	1.18%	1.78%	1.28%	0.51%	4.15%	3.03%	1.12%
Markov	3.33%	2.15%	1.18%	4.26%	3.75%	0.51%	9.97%	8.85%	1.12%
CSG	2.08%	0.90%	1.18%	4.72%	4.21%	0.51%	6.19%	5.06%	1.12%

Abbildung 4: Fehlerhafte Klassifizierung für die drei Verfahren

Das Product Distribution Modell erzielt durchweg annehmbare Ergebnisse und produzierte deutlich weniger Fehler als die beiden konkurrierenden Verfahren.

Protocol		Product			Markov			CSG		
	%	Err.%	Prec.%	Rec.%	Err.%	Prec.%	Rec.%	Err.%	Prec.%	Rec.%
① DNS	26.28	0.09	99.94	99.78	0.61	97.89	99.97	0.45	98.82	99.52
HTTP	12.24	0.07	100.00	99.99	0.09	100.00	99.98	0.74	99.91	99.99
NBNS	44.89	0.35	100.00	99.25	0.40	99.82	99.31	0.17	99.71	99.99
NTP	5.29	0.00	100.00	100.00	1.19	99.96	77.84	0.25	99.83	95.65
SSH	0.22	0.14	68.39	100.00	1.10	17.39	100.00	0.05	99.22	100.00
② DNS	23.14	0.04	99.88	99.93	0.29	98.88	99.99	1.97	94.37	97.59
HTTP	0.67	0.27	76.02	97.54	0.09	90.68	99.93	0.22	76.87	99.38
NBNS	6.94	0.00	100.00	100.00	1.96	78.06	100.00	0.81	90.34	99.97
NTP	0.57	0.01	99.95	99.72	0.51	100.00	11.29	0.40	86.65	48.76
SSH	0.44	0.17	75.28	100.00	0.00	99.63	100.00	0.00	99.99	100.00
③ DNS	54.78	0.26	99.90	99.95	1.90	97.13	99.98	1.43	98.47	99.15
HTTP	9.17	0.38	97.46	99.62	0.33	97.21	99.72	1.21	95.14	97.19
NBNS	7.03	0.01	100.00	99.81	1.25	85.66	99.81	0.33	96.04	99.45
NTP	6.70	0.02	99.99	99.94	5.39	78.07	29.61	0.36	99.82	96.58
SSH	0.08	0.08	68.81	81.82	0.09	0.00	0.00	0.03	95.40	82.01

Abbildung 5: Fehler-, Präzisions- und Recall-Rate der ausgewählten Protokolle. Die 2. Spalte zeigt den Gesamtanteil des Protokolls. Benutzte Datensätze:(1)Cambridge (226,046 flows), (2)Wireless (403,752 flows), (3)Departmental (1,064,844 flows).

6 Zusammenfassung

Die Ergebnisse zeigen, dass eine selbstständige Identifikation von Applikationen unabhängig von Portnummern möglich ist. Es werden drei Verfahren vorgestellt, die unterschiedliche Ergebnisse hervorbringen. Product Distribution vergleicht statistisch die Bytes in den Nutzdaten, Markov Process betrachten statistisch Byte-Übergänge und Common Substring Graphs betrachten Teil-Strings in den Nutzdaten. Die vorgestellten Verfahren laufen nicht auf allen Protokollen ohne gröbere Fehler. Einerseits ist dies den Charakteristiken der Protokolle geschuldet, andererseits muss aber auch die Modellierung von Protokollen weiterhin verbessert werden, damit optimale Ergebnisse erzielt werden.

Literatur

[1] Google Page Rank,
http://google.com/pagerank, 13.06.2007.

[2] P. Haffner, S. Sen, O. Spatscheck, and D. Wang. ACAS: Automated construction of application signatures. In *Proceedings of the 2005 Workshop on Mining Network Data*, pages 197-202, 2005.

[3] Internet Assigned Numbers Authority. TCP am UDP Port Numbers (IANA)
http://www.iana.org/assignments/port-numbers, 13.06.2007.

[4] J. Ma, K. Levchenko, C. Kreibich, S. Savage, and G. Voelker. Unexpected Means of Protocol Inference. In *Proceedings of the 2006 Internet Measurement Conference*, 2006.

[5] NIST/SEMATECH e-Handbook of Statistical Methods,
http://www.itl.nist.gov/div898/handbook/, 13.06.2007.

[6] T. F. Smith and M. S. Waterman. Identification of Common Molecular Subsequences. In *Journal of Molecular Biology*, 147, 1981. http://gel.ym.edu.tw/c̃hc/AB_papers/03.pdf, 13.06.2007.

[7] TRANSMISSION CONTROL PROTOCOL. In RFC: 793, 1981
http://tools.ietf.org/html/rfc793, 13.06.2007.

[8] Robert J. Tibshirani, Trevor J. Hastie. The Elements of Statistical Learning: Data Mining, Inference, and Prediction, pages 520, 2005. Springer.

[9] Andrew S. Tanenbaum. Computer Networks, Forth Edition, pages 383, 2003. Prentice Hall.

[10] Wireshark,
http://www.wireshark.org, 13.06.2007.