Filtern von Spam-Nachrichten mit kontextfreien Grammatiken

Philipp Trinius[1] Felix C. Freiling*[2]

[1] Universität Mannheim, Deutschland
[2] Friedrich-Alexander-Universität Erlangen-Nürnberg

Abstract: Spam wird heute überwiegend mittels so genannter musterbasierte Spam-Algorithmen über Botnetze verteilt. Bei musterbasiertem Spam werden die Spam-Nachrichten erst von den Bots aus einem Muster (*template*) und Fülldaten zusammengesetzt. Filteransätze für musterbasierten Spam versuchten bisher, dieses Muster aus den abgefangenen Nachrichten zu extrahieren und auf reguläre Ausdrücke abzubilden. Diese Technik kann aber durch die Umsortierung von Worten oder Zeilen leicht umgangen werden. Wir schlagen einen neuartigen Filteransatz vor, der auf kontextfreien Grammatiken basiert. Unser Ansatz lernt dabei nicht die Muster sondern die „Inhalte" der Nachrichten. Das Resultat ist eine Grammatik, die zum Filtern von Nachrichten aus einer spezifischen Spam-Kampagne verwendet werden kann. Die Filterergebnisse dieses Ansatzes sind sehr gut: Teilweise erreichen aus einer einzelnen Nachricht erstellte Filter bereits Erkennungsraten von über 99 Prozent.

1 Einführung

Der überwiegende Anteil an Spam wird aktuell über Botnetze versendet. Die verschickten Nachrichten werden dabei erst von den Bots selbst, aus vordefinierten Mustern für die einzelnen Spam-Kampagnen und *Fülldaten* wie Empfänger- und Absenderlisten, zusammengesetzt. Die verwendeten Muster bieten hierbei einen Ansatzpunkt für Kampagnenspezifische Filter. Bei den bisher veröffentlichen Ansätzen wurde jeweils versucht, das Muster zu approximieren und zum Filtern der Nachrichten zu verwenden. Dazu werden die Spambots in sogenannten Sandnets ausgeführt und aus den verschickten Nachrichten wird, über einen *Longest Common Substring* (LCS) Algorithmus, das Template berechnet. Die aus den abgegriffenen Nachrichten generierten Filter passen im besten Fall nur auf Nachrichten der Kampagne, das heißt, der Filter liefert keine *false positives*.

Die auf LCS-Berechnungen basierenden proaktiven Filteransätze versuchen, das Template als regulären Ausdrück darzustellen. Dieser enthält nur die Teile der Nachrichten, die in allen Nachrichten enthalten sind. Dem Ansatz liegt damit die Annahme zugrunde, dass das in der Kampagne verwendete Template starr ist und die resultierenden Spam-Nachrichten sich nur an einzelnen Stellen unterscheiden. Sobald das Template nicht nur Ersetzungen, sondern auch Permutationen von Wörtern, Sätzen, oder auch ganzen Textblöcken zulässt,

*Kontaktautor, Kontaktadresse: Am Wolfsmantel 46,91058 Erlangen, Deutschland.

ist die Beschreibung über einen regulären Ausdruck nur bedingt möglich. Um die Permutationen in dem extrahierten Muster abzubilden, müssen sehr komplexe reguläre Ausdrücke konstruiert werden. Zudem muss immer sichergestellt werden, dass es sich bei den aufgenommenen Informationen (Varianten) nicht um Fülltexte handelt.

Bisherige Arbeiten. Kreibich et al. [KKL+08] veröffentlichten mit ihrer Studie zum STORM WORM Botnetz die erste Arbeit, die das Verfahren des musterbasierten Spam für Botnetze im Detail beschreibt. Die Möglichkeit, die von den Bots verwendeten *Templates* bedingten Regelmäßigkeiten innerhalb der Nachrichten zum Filtern zu verwenden, wurde erstmals von Stern [Ste08] erwähnt, dort aber noch nicht umgesetzt.

In der Folge entstanden einige Arbeiten [XYA+08, JMGK09, GHT09], die den von Spambots generierten Spam zur Extraktion Kampagnen-spezifischer Informationen heranzuziehen. Während Xie et al. [XYA+08] und John et al. [JMGK09] lediglich die in den Nachrichten enthaltenen URLs herauslesen, extrahieren Göbel, Holz und Trinius [GHT09] das Muster der Kampagne selbst aus den Nachrichten und bilden es in einen regulären Ausdruck ab. Pitsillidis et al. [PLK+10] erweitern den Ansatz von Göbel, Holz und Trinius [GHT09] um *Dictionaries* für ausgewählte, variable Textpassagen, die mit in die regulären Filterausdrücke einfließen.

Ansatz und Ergebnisse. Der von uns vorgestellte Filteransatz löst sich von der starren Beschreibung des Templates als regulären Ausdruck. Statt das Template aus den abgegriffenen Nachrichten zu extrahieren, wird in unserem Ansatz der Inhalt der innerhalb der Kampagne verschickten Nachrichten gelernt. Aus diesen Daten wird eine kontextfreie Grammatik konstruiert, die zum Filtern von Spam-Nachrichten verwendet werden kann. Hinter diesem Ansatz steht die Annahme, dass Spammern nur eine beschränkte Anzahl an Wörtern und Sätzen zum Beschreiben ihrer Produkte zur Verfügung stehen. Der hier vorgestellte Ansatz fasst Spam-Nachrichten also weniger als Kette von Zeichen auf sondern als in einer *natürlichen Sprache* formulierte Texte.

Die Evaluierung der sprachbasierten Filteransätze zeigt, dass sich die Nachrichten musterbasierter Spam-Kampagnen aus nur wenigen charakteristischen Wörtern und Sätzen zusammensetzen. Der Großteil dieser Daten ist in allen Nachrichten einer Kampagne enthalten und damit schon nach der Analyse einzelner oder weniger Nachrichten bekannt. Der Filteransatz benötigt daher sehr wenige Nachrichten, um sehr effektive Filter für die einzelnen Kampagnen zu berechnen. Für die zur Evaluierung herangezogenen Spam-Läufe konnten teilweise schon nach dem Lernen einer einzelnen Spam-Nachricht Erkennungsraten von über 99 Prozent erreicht werden. Dieser Beitrag stellt die Ergebnisse in aller Kürze vor. Für weitere Evaluationen und technische Details sei auf Trinius [Tri11] verwiesen.

Das Dokument gliedert sich wie folgt: Wir stellen zunächst den Systemaufbau zum Abgreifen und Analysieren der Spam-Nachrichten vor (Abschnitt 2). Anschließend werden in Abschnitt 3 die Nachteile der auf regulären Ausdrücken basierenden Filter diskutiert. Sie dienen gleichzeitig als Motivation für den ebenfalls in diesem Abschnitt vorgestellten, auf kontextfreien Grammatiken basierenden Filteransatz. Die Evaluation der auf diesem Ansatz generierten Filter erfolgt in Abschnitt 4. Das Dokument schließt mit einem Aus-

blick.

2 Systemaufbau

Abbildung 1 zeigt den Aufbau eines um einen lokalen Mail-Server und den Filtergenerator erweiterten Sandnets. Mit Sandnet bezeichnen wir dabei Analysesysteme, mit deren Hilfe die Netzwerkkommunikation von Schadprogrammen überwacht und analysiert werden kann. Die von dem Schadprogramm zur Laufzeit lokal auf dem Hostsystem ausgeführten Operationen, werden nicht betrachtet. Alle von den Bots verschickten Spam-Nachrichten werden an dem zwischen den einzelnen Analysesystemen und dem Internet platzierten Gateway auf einen lokalen Mail-Server umgeleitet. Dieser übergibt die Nachrichten direkt an den Filtergenerator, oder speichert sie im Dateisystem ab.

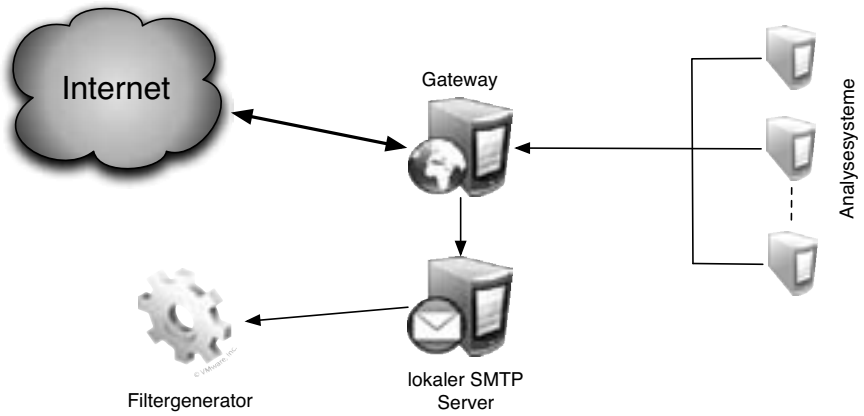

Abbildung 1: Systemaufbau zur Analyse von Spambots.

3 Filtern mit kontextfreien Grammatiken für E-Mail-Nachrichten

Die auf LCS-Algorithmen und regulären Ausdrücken basierenden Spam-Filter stoßen relativ schnell an ihre Grenzen. Schon das Vertauschen von zwei aufeinanderfolgenden Wörtern in zwei untersuchten Nachrichten führt entweder zu dem Verlust eines der Worte, oder muss über Oder-Verknüpfungen in den regulären Ausdruck abgefangen werden. Dieses Problem wird bei Permutationen in Listen, oder zwischen ganzen Sätzen noch größer. Obwohl sich der „Inhalt“ der Nachrichten nicht unterscheidet, verlieren die generierten Spam-Filter an Genauigkeit.

Der in dieser Arbeit vorgestellte Filteransatz löst diese Probleme, indem er eine mächtige-

re Sprache zur Beschreibung der Spam-Kampagnen einsetzt. Statt das Template der Kampagne in einem regulären Ausdruck auszudrücken, kommt eine kontextfreie Grammatik für E-Mail-Nachrichten zum Einsatz, die um die Kampagnen-spezifischen Informationen, dieses sind die verwendeten Wörter und Sätze, erweitert wird.

3.1 Grundgrammatik für E-Mail-Nachrichten

Als Ausgangspunkt für das Lernen des Spam-Templates verwenden wir die in Listing 1 dargestellte Grundgrammatik. Diese besteht aus acht Ableitungsregeln und fünf Token. Der NEWMAIL-Token beschreibt dabei die optionale Absender-Zeile vor dem eigentlichen Header. Ein HEADER-Token passt auf jedes beliebige Feld des Headers, inklusive dem das Feld abschließenden Zeilenumbruch. Dabei werden auch leere oder sich über mehrere Zeilen erstreckende Felder von genau einem Token abgedeckt. Der NEWLINE-Token stellt einen einzelnen Zeilenumbruch dar und ist notwendig, um den Wechsel zwischen dem E-Mail-Header und dem -Body zu erkennen. Der Token ist nur beim Parsen des Headers aktiv. Im Body der Nachricht werden die Zeilenumbrüche ignoriert, da sie keinerlei Bedeutung für die enthaltenen Sätze besitzen. Über den WORD-Token wird jede beliebige Zeichenfolge abgedeckt und Satzzeichen werden durch den PUNCTUATION-Token dargestellt. Da beim Parsen der Nachrichten die enthaltenen Sätze im Vordergrund stehen, müssen die Satzzeichen, die diese voneinander trennen, bewusst verarbeitet werden.

```
spammail        : NEWMAIL header NEWLINE body
                | header NEWLINE body
header          : header_fields
header_fields   : HEADER header_fields
                | HEADER
                | empty
body            : sentences
sentences       : sentences sentences
                | sentence sentences
                | sentence PUNCTUATION
sentence        : words
words           : WORD words
                | WORD
                | empty
empty           :
```

Listing 1: Die Ableitungsregeln der Grundgrammatik für E-Mail-Nachrichten.

Diese einfache Grammatik genügt zwar, um jede E-Mail-Nachricht abzuleiten, ist aber zum Lernen von Inhalten noch zu unpräzise. Alle Felder des Headers werden durch einen Token repräsentiert, der nicht zwischen den Bezeichnern und dem Wert des Feldes unterscheidet. Mögliche Dateianhänge – diese werden als *base64*-codierter Text in den Body der Nachricht integriert – werden als Wörter gelernt und E-Mail-Adressen und URLs werden an den „Satzzeichen“ zerlegt und ebenfalls wortweise gelernt. Um diese Fehlinterpre-

tationen zu unterbinden und auch Informationen aus einem Header-Feld extrahieren und lernen zu können, sind zusätzliche Token und Grammatikregeln notwendig.

Aus dem Header sind (momentan) lediglich die Felder *Subject* und *Content-Type* von Interesse [GHT09]. Zur Verarbeitung der beiden Felder wird die Grammatik um zwei entsprechende Token und die diese einbindenden Grammatikregeln erweitert. Das SUBJECT-Token passt dabei ausschließlich auf den Feldbezeichner. Der Wert des Feldes wird über die zusätzliche Grammatikregel als Satz oder Sätze, gefolgt von einem NEWLINE-Token, definiert. Der Wert des *Subject*-Felds wird damit genauso behandelt wie jeder beliebige Satz aus dem Nachrichten-Body. Da sich das *Subject*-Feld und Sätze des Bodies gerade bei Spam-Nachrichten vielfach gegeneinander austauschen lassen, macht eine gesonderte Betrachtung des *Subject*-Feldes wenig Sinn. Das neue `subject`-Symbol wird durch die veränderte Regel zum Ableiten des `header`-Symbols in die Grammatik integriert.

Der *Content-Type* ist das zweite interessante Header-Feld. Es wird ausgewertet, um ein eventuell enthaltenes *Boundary*-Attribut auszulesen und zu überprüfen, ob die E-Mail-Nachricht HTML-codiert ist. Anhand der *Boundary* lassen sich die einzelnen Teile der Nachrichten erkennen und voneinander trennen. Wurde ein entsprechender Token gefunden, wird der Nachrichten-Body beim Übergang des Lexer vom Header zum Body wie folgt optimiert: Aus den HTML-codierten Blöcken werden alle HTML-Tags entfernt, wobei darin enthaltenen URLs erhalten bleiben. Des Weiteren werden *base64*-codierte Blöcke durch ihren MD5-Hashwert ersetzt. Dadurch wird zum einen verhindert, dass unnötige Wörter aus dem „Multi-Part Header“ oder Dateianhängen gelernt werden und zum anderen können die in der Nachricht enthaltenen Dateianhänge – als MD5-Hashwert – selbst gelernt werden.

Neben dem Dateianhang kommt den im Nachrichten-Body eventuell enthaltenen E-Mail-Adressen und URLs eine besondere Rolle beim Filtern der Nachrichten zu. Um diese erkennen zu können, wird die Grammatik um zwei Token zum Erkennen von E-Mail-Adressen und URLs erweitert, über die sich die in den Nachrichten enthaltenen URLs und E-Mail-Adressen während des Parsens lernen lassen.

Die erweiterte Grundgrammatik besitzt neben den hier vorgestellten Erweiterungen noch zusätzliche Token und Regeln, zum Beispiel um Zahlen erkennen zu können, auf die an dieser Stelle nicht näher eingegangen wird. Insgesamt besteht die erweiterte Grammatik aus 14 Token und 25 Grammatikregeln [Tri11].

3.2 Lernen der Kampagnen-spezifischen Informationen

Der Prototyp zum Parsen, Lernen und anschließenden Filtern von E-Mail-Nachrichten ist in PYTHON implementiert und verwendet die von dem Modul PLY [Bea09] bereitgestellte LEX/YACC-Implementierung. Um das Lernen von neuen Wörtern, E-Mail-Adressen, URLs, Dateianhängen und den aus diesen Teilen zusammengesetzten Sätzen zu ermöglichen, wurden selbstmodifizierende Klassen für den Lexer und Parser implementiert. Diese modifizieren ihren eigenen Quellcode und sind so in der Lage neue Token und Regeln zu lernen, die nach einem erneuten *Import* der erweiterten Klassen auch in die, dem Par-

ser zugrunde liegende, Grammatik eingehen. Die zu lernenden Informationen werden im Folgenden allgemein als *Daten* bezeichnet.

In der Regel verschicken Spambots während der Analyse E-Mail-Nachrichten aus verschiedenen Spam-Kampagnen [GHT09]. Das heißt, für die abgefangenen Nachrichten müssen mehrere Filter (Grammatiken) gelernt werden. Eine alle Kampagnen umfassende Grammatik wäre zu generisch und damit letztlich unbrauchbar. Um alle Kampagnen optimal abzudecken, muss daher für jede abgegriffene E-Mail-Nachricht entschieden werden, welcher Kampagne sie zuzuordnen ist. Dem System sind beim Lernen einer neuen Grammatik weder die letztlich die Spam-Kampagnen abdeckende Grammatik, noch die zum Parsen der verarbeiteten Nachricht notwendigen Token und Regeln bekannt. Somit muss die Nachricht immer erst vollständig gelernt werden, bevor entschieden werden kann, ob die Nachricht zu der momentan bearbeiteten Spam-Kampagne gehört.

Der Lernalgorithmus läuft in 5 Schritten ab (siehe Abbildung 2):

1. Von den Basisklassen *SpamLexer* und *SpamParser* werden zwei Klassen (*PrototypeLexer* und *ProtoypeParser*) abgeleitet, die die Grundgrammatik definieren.

2. Die erste Nachricht wird dem Datensatz entnommen und dem *ParserGenerator* übergeben. Dieser importiert die Klassen *PrototypeLexer* und *ProtoypeParser* und erzeugt ein Lexer/Parser-Paar. Beim Parsen der Nachricht wird die Grammatik solange erweitert, bis die Nachricht fehlerfrei abgeleitet werden kann. Die um die Token und Regeln erweiterten Klassen *PrototypeLexer* und *ProtoypeParser* definieren die Grammatik $grammar_1$.

3. Anschließend wird die nächste Nachricht dem Datensatz entnommen und die beiden Prototyp-Klassen werden kopiert. Der *ParserGenerator* importiert die Kopien der beiden Prototyp-Klassen und erzeugt ein Lexer/Parser-Paar. Die von diesen definierte Grammatik wird solange um Token und Regeln erweitert, bis die Nachricht fehlerfrei verarbeitet werden kann. Die erzeugte Grammatik wird mit $grammar_2$ bezeichnet.

4. Die beiden Grammatiken $grammar_1$ und $grammar_2$ werden miteinander verglichen. Dabei wird bestimmt, wie viel Prozent der beim Ableiten der Nachricht verwendeten Token und Regeln schon in $grammar_1$ definiert sind (α_t, α_r) und wie viel Prozent der in $grammar_1$ definierten Token und Regeln beim Parsen angewendet wurden (β_t, β_r). Unterschreiten alle Werte $\alpha_t, \alpha_r, \beta_t$ und β_r eine Schranke ϵ, gehört die Nachricht nicht zu der aktuell bearbeiteten Spam-Kampagne und die Nachricht wird der Liste verworfener Nachrichten hinzugefügt.

   Liegt eines der Wertepaare α, β für die Token oder Regeln über der Schranke ϵ, wird die Nachricht als zugehörig eingestuft und die Klassen *PrototypeLexer* und *ProtoypeParser* werden mit ihren in Schritt 3 erweiterten Kopien überschrieben.[1] Der Algorithmus wird anschließend in Schritt 3 fortgesetzt.

[1] Der optimale Wert für ϵ wurde noch nicht bestimmt. Die hier präsentierten Ergebnisse wurden mit $\epsilon = 0,8$ generiert. D.h. eine Nachricht wird nur dann als zugehörig eingestuft, wenn entweder mindestens 80% der in $grammar_1$ enthaltenen Token und Regeln angewendet worden, um die Nachricht abzuleiten, oder falls 80% der in $grammar_2$ enthaltenen Token und Regeln bereits in $grammar_1$ enthalten sind.

5. Nachdem der komplette Datensatz einmal durchlaufen ist, ist die Generierung des Parsers abgeschlossen. Die die gesuchte Grammatik beschreibenden Prototyp-Klassen werden gesichert und stellen die Basis des kontextfreien Filters dar. Anschließend wird der Algorithmus auf den verworfenen Nachrichten neu gestartet.

Das Lernen einer neuen Grammatik wird immer durch Manipulationen am Quellcode, der die Grammatik definierenden Klassen, umgesetzt. Das Lernen neuer Daten wird entweder nach dem erfolgreichen Parsen einer Nachricht, oder aus der Fehlerbehandlungsroutine heraus angestoßen: Der erste Fall tritt ein, wenn bei der Verarbeitung der Nachricht ein neuer Satz aus noch unbekannten Token geparsed wird. Der Parser kann diesen ohne Fehler aus den generischen Token (WORD, UNIT, E-MAIL, etc.) und Regeln (`word`, `words`, `sentence`, etc.) ableiten. Anschließend muss die Erweiterung der Grammatik um neue Token und die dem Satz entsprechende Grammatikregel angestoßen werden. Wenn beim Parsen neben unbekannten Token auch bekannte Token innerhalb eines Satzes abgeleitet werden, tritt hingegen ein Fehler auf. In der Lernphase dürfen bekannte Token nur innerhalb der definierten Grammatikregeln geparsed werden. Erst diese Einschränkung des Parsers macht das Lernen von Modifikationen in bekannten Sätzen möglich. Die Fehlerbehandlungsroutine untersucht dabei den aktuellen Parserzustand und versucht den Fehler aufzulösen. Die aktuelle Implementierung kann innerhalb eines Satzes einen der folgenden „Fehler“ abfangen und auflösen, ohne dass das Parsen der Nachricht neu angestoßen werden muss: ein zusätzliches Wort, eine Vertauschung zweier aufeinander folgender Wörter, ein ausgelassenes Wort und ein ersetztes Wort. Liegt einer dieser Fehler vor, wird die Modifikation des Satzes in die Grammatik aufgenommen und der Parser setzt die Bearbeitung hinter dem Satz fort.

3.3 Filtern mit LEX und YACC

Die *SpamFilter* bekommen die zu filternden Nachrichten und die generierten Prototypen für den *SpamLexer* und den *SpamParser* übergeben. Während der Lexer unverändert eingesetzt werden kann, muss der Parser noch um Grammatikregeln zum Ableiten der einzelnen Token erweitert werden. Die zusätzlichen Grammatikregeln sind notwendig, damit sich beim Parsen der Nabchrichten auch unbekannte, aber aus bekannten Wörtern bestehende Sätze ableiten lassen.

4 Evaluierung der *kontextfreien* Spam-Filter

Um die Effektivität der auf kontextfreien Grammatiken beruhenden Spam-Filter zu evaluieren, wurden Spam-Läufe aus drei Botnetzen herangezogen, wovon wir die Ergebnisse auf 437 Läufen des STORM-Botnetz hier vorstellen. Ein Spam-Lauf umfasst dabei alle von dem Bot whrend einer Ausfhrung im Sandnet verschickten E-Mail Nachrichten. Als Vergleichswerte wurden die Filterergebnisse der auf regulären Ausdrücken basierenden Spam-Filter [GHT09] verwendet. Basierend auf den Filterergebnissen können die beiden

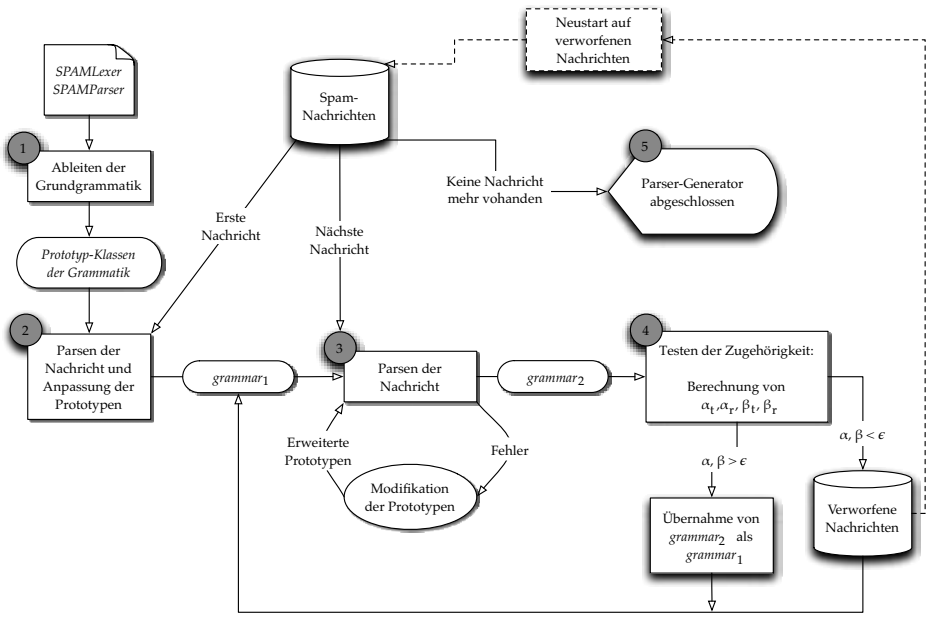

Abbildung 2: Schematischer Ablauf des Lernens. Die fünf Schritte des Algorithmus sind farblich markiert.

Ansätze damit direkt gegenübergestellt und miteinander verglichen werden.

Die Filter wurden jeweils für Teilmengen der Nachrichten, das heißt, für jeweils 0,01 Prozent, 0,1 Prozent, einem Prozent, 10 Prozent und 100 Prozent der Nachrichten, erzeugt und anschließend zum Filtern der Gesamtdaten verwendet. Dieser Prozess wurde für jeden Datensatz und jede Kampagne bis zu 10 mal wiederholt, um stabile Ergebnisse zu erhalten.

Der hier vorgestellte Filteransatz ist sehr rechenintensiv. Das stetige Anpassen der Grammatik während der Lernphase macht die Filtergenerierung langsam und lässt sich nicht parallelisieren. Deshalb ist es umso wichtiger, dass der Filter-Generierungsprozess nur sehr wenige Nachrichten benötigt, um Filter zu berechnen, die die gesamte Spam-Kampagne abdecken, ohne dabei *false-positives* zu verursachen. Letzteres wird durch die starke Spezifizierung der Spam-Filter auf einzelne Kampagnen sichergestellt. Die Filter bestehen aus einer Kampagnene-spezifischen Grammatik, die die innerhalb der Kampagne verwendeten Wörter und Sätze enthält und Nachrichten auf diese hin untersucht. Jede Nachricht, die von einem Filter als zugehörig klassifiziert wurde, muss sich fehlerfrei mit der spezifischen Grammatik ableiten lassen und dabei zu mindestens 80% aus den Kampagnenspezifischen Wörtern und Sätzen bestehen oder mindestens 80% aller für die Kampagne gelernten Wörter und Sätze enthalten.

Aus dem STORM-Botnetz wurden 437 jeweils 5.000 E-Mail-Nachrichten umfassende Spam-Läufe zur Evaluierung des Filteransatzes herangezogen. Die durchschnittlich über alle 437 Spam-Läufe erreichten Erkennungsraten sind in Tabelle 1 dargestellt. Zusätzlich wurde ein rund 10.000 Nachrichten umfassender Spam-Datensatz herangezogen, um die Filter auf *false positives* hin zu untersuchen. Der Datensatz wurde jeweils einem Spam-Filter [2] pro Spam-Lauf vorgelegt. Jeder der 437 Filter kategorisierte die Nachrichten dabei als nicht-zugehörig, d.h., die Filter lieferten auf dem Datensatz keine *false positives*.

Lerndatensatz		**Filterergebnisse**		
# E-Mails	*in %*	*Kampagnen**	*Erkennungsrate (LexYacc)*	*Erkennungsrate (RegEx)*
1	0,01	1	99,7578 %	1,0567 %
5	0,1	1	99,5373 %	32,5692 %
50	1,0	1	99,0036 %	92,5036 %
500	10	1	99,9997 %	99,9936 %
5000	100	1	100,0000 %	100,0000 %

Tabelle 1: Filterergebnisse für das STORM-Botnetz. Die Angaben sind Durchschnittswerte über alle 437 Spam-Läufe und die 10 pro Lauf und Teilmenge erzeugten Filter. (* Die Anzahl an Kampagnen bezieht sich jeweils auf einen Spam-Lauf.)

Der Filteransatz liefert mit über 99 Prozent Erkennungsrate durchweg sehr gute Ergebnisse. Schon der auf einer einzigen Nachricht basierende Filter ist gut genug, um die Spam-Kampagnen zuverlässig zu filtern. Für die auf regulären Ausrücken basierenden Filter liegt die durchschnittliche Erkennungsrate nach dem Lernen einer Nachricht noch

[2] Mit den auf 0,01 Prozent der Nachrichten erzeugten Filtern wurden hierbei die am wenigsten genauen Filter herangezogen.

bei rund einem Prozent. Erst Filter, die auf 50 Nachrichten basieren, können ähnlich hohe Erkennungsraten aufweisen, auch wenn der Durchschnitt für diese Filter mit 92,5 Prozent immer noch bedeutend schlechter ist.

Aus welchem Grund die Erkennungsrate für die aus 5 und 50 Nachrichten gelernten Filter geringer ist, als die der nur auf einer Nachricht basierenden Filter, kann aufgrund der großen Anzahl an Filter nicht zweifelsfrei geklärt werden. Erste Analysen zeigen, dass die schlechteren Ergebnisse nicht auf einzelne Ausreißer mit sehr schlechten Erkennungsraten zurückzuführen sind, sondern dass viele der Filter leicht schwächere Ergebnisse liefern. Das Gesamtergebnis wird damit von sehr vielen Filtern beeinflusst, was eine detaillierte Untersuchung zusätzlich erschwert.

Die auf einer Grammatik basierenden Filter bestehen, stark vereinfacht, aus gelernten Worten und Sätzen. Diese beschreiben, wie die während der Filtergenerierung bearbeiteten E-Mail-Nachrichten aufgebaut sind. Das heißt, der Filter kennt genau die Wörter und Sätze, die ihm zum Lernen vorgelegt wurden. In Tabelle 2 sind die Anzahl der im Durchschnitt von den Filtern gelernter Wörter und Sätze aufgelistet.

Lerndatensatz	**Filterdaten**	
# E-Mails	*# gelernter Wörter*	*# gelernter Sätze*
1	112,28	23,07
5	123,83	28,09
50	189,26	65,93
500	231,54	98,61
5.000	231,56	98,63

Tabelle 2: Anzahl der im Mittel gelernten Wörter und Sätze für die 437 im STORM-Botnetz generierten Spam-Läufe.

Aus den gelernten Daten lässt sich ableiten, dass eine E-Mail-Nachricht der analysierten Spam-Läufe durchschnittlich aus 112 Wörtern und 23 Sätzen besteht und dass die innerhalb einer Kampagne verschickten Nachrichten zusammen aus rund doppelt so vielen Wörtern, und fast viermal so vielen Sätzen bestehen. Da schon die auf nur einer Nachricht generierten Filter eine Erkennungsrate von über 99 Prozent aufweisen, muss innerhalb der Spam-Kampagnen jeweils eine Grundmenge von Wörtern und Sätzen existieren, die in jeder E-Mail-Nachricht einer Kampagne enthalten sind. Die nur langsam steigende Anzahl an Wörtern und Sätzen weist zudem darauf hin, dass die Nachrichten neben der Grundmenge nur sehr wenige zusätzliche Wörter und Sätze enthalten. Das Vorhandensein der Grundmenge und die geringe Varianz durch zusätzliche Wörter und Sätze kommen dem auf Grammatiken basierten Filteransatz entgegen und erklären die für das STORM-Botnetz erreichten, sehr guten Ergebnisse.

5 Ausblick

Das Ziel des hier vorgestellten Filteransatzes war es, die Analysemöglichkeiten natürlicher Sprache auf Basis von kontextfreien Grammatiken zu untersuchen. Dabei kommen wir zu einem durchaus differenzierten Bild: Während die Erstellung regulärer Ausdrücke deutlich schneller ist als das Lernen einer kontextfreien Grammatik, so steigen die Erkennungsraten der kontextfreien Filter schneller mit der Zahl der eingegangenen Spam-Nachrichten als die der regulären Filter. Wenn es also darum geht, mit dem Lernen von nur wenigen Nachrichten gute Filterergebnisse zu bekommen, dann bestehen zur Zeit kaum bessere Alternativen zum hier vorgeschlagenen Ansatz.

Auch wenn der vorgestellte Ansatz bereits sehr gute Ergebnisse liefert, reizt die Grammatik die sich bietenden Möglichkeiten noch lange nicht aus. Beispielsweise ließen sich über entsprechende Token und Grammatikregeln sprach-spezifische Grammatiken erzeugen, die die in einer E-Mail-Nachricht enthaltenen Sätze nicht nur als eine Aneinanderreihung von gleichartigen Token definiert, sondern auch den Aufbau der Sätze beachtet. Dadurch könnte der Inhalt der Kampagne noch besser erfasst und die erlaubte Varianz genauer eingestellt werden. Auch aus diesem Blickwinkel macht ein möglicher Übergang zur mächtigeren Klasse der kontext-sensitiven Grammatiken als Basis des Systems aus unserer Sicht vorerst keinen Sinn.

Danksagung

Wir danken Andreas Pitsillidis für die Bereitstellung seiner Spam-Datensätze und Tilo Müller für seine wertvollen Hinweise zu dieser Arbeit.

Literatur

[Bea09] David M. Beazley. PLY (Python Lex-Yacc), 2009. URL: `http://www.dabeaz.com/ply/ply.hmtl`.

[GHT09] Jan Göbel, Thorsten Holz und Philipp Trinius. Towards Proactive Spam Filtering. In *Proceedings of Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA)*, Seiten 38–48, 2009.

[JMGK09] John P. John, Alexander Moshchuk, Steven D. Gribble und Arvind Krishnamurthy. Studying Spamming Botnets Using Botlab. In *Proceedings of NSDI'09*, 2009.

[KKL+08] Christian Kreibich, Chris Kanich, Kirill Levchenko, Brandon Enright, Geoffrey M. Voelker, Vern Paxson und Stefan Savage. On the Spam Campaign Trail. In *Proceedings of LEET'08*, 2008.

[PLK+10] A. Pitsillidis, K. Levchenko, C. Kreibich, C. Kanich, G.M. Voelker, V. Paxson, N. Weaver und S. Savage. Botnet Judo: Fighting Spam with Itself. In *Proceedings of the 17th Annual Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, USA, March 2010.

[Ste08] Henry Stern. A Survey of Modern Spam Tools. In *Proceedings of the 5th Confernece in Email and Anti-Spam*, 2008.

[Tri11] Philipp Trinius. *Musterbasiertes Filtern von Schadprogrammen und Spam*. Dissertation, Universität Mannheim, 2011.

[XYA+08] Yinglian Xie, Fang Yu, Kannan Achan, Rina Panigrahy, Geoff Hulten und Ivan Osipkov. Spamming Botnets: Signatures and Characteristics. In *Proceedings of SIGCOMM'08*, 2008.