Systematische Ableitung von Signaturen durch Wiederverwendung am Beispiel von *Snort*

Sebastian Schmerl, René Rietz, Hartmut König

BTU Cottbus
LS Rechnernetze und Kommunikationssysteme
03013 Cottbus, Postfach 10 13 44
(sbs, rrietz, koenig)@informatik.tu-cottbus.de

Abstract: Die Wirksamkeit von Intrusion Detection Systemen mit Signaturanalyse hängt entscheidend von der Präzision der verwendeten Signaturen ab. Die Ursachen unpräziser Signaturen sind hauptsächlich der Signaturableitung zuzuschreiben. Die Spezifikation einer Signatur ist aufwendig und fehleranfällig. Methoden für ein systematisches Vorgehen existieren bisher kaum. In diesem Papier stellen wir einen Ansatz zur systematischen Ableitung von Signaturen durch Wiederverwendung von Signaturen bzw. Signaturfragmenten vor, der ursprünglich für Multi-Step-Attacken entwickelt wurde. Wir zeigen, dass er auch für Single-Step-Attacken genutzt werden kann. Dazu verwenden wir *Snort*-Signaturen.

Keywords: Intrusion Detection, Signaturanalyse, Signature Engineering, systematische Signaturableitung, Single-Step-Attacken, Signatursprachen, Snort

1 Einleitung

Um den Gefahren eines sich ständig vergrößernden Bedrohungspotentials zu begegnen, werden in zunehmendem Maße Intrusion Detection Systeme (IDS) eingesetzt. In der praktischen Anwendung erweist sich dabei bisher die Signaturanalyse als die effizientere Form. Sie steht hier im Mittelpunkt der Betrachtung. Signaturbasierte Analyseverfahren untersuchen Protokoll- bzw. Auditdaten nach Mustern bekannter Sicherheitsverletzungen, den so genannten *Signaturen*. Die Wirksamkeit der Analyse hängt entscheidend von der Präzision der verwendeten Signaturen ab. Unpräzise Signaturen schränken die Erkennungsfähigkeit der Analyse stark ein und führen zu den typischen hohen Fehlalarmraten. Die Ursachen dafür sind fast ausschließlich auf der Ebene der Signaturableitung zu suchen. Insbesondere das Ableiten von Signaturen aus vorliegenden Angriffsszenarien erweist sich häufig als Schwachpunkt. Dies erfolgt zumeist empirisch auf der Grundlage des Wissens und der Erfahrung von Experten. Verfahren für eine systematische Ableitung von Signaturen gibt es bisher kaum. In [1] wird ein erster Ansatz für allgemeingültiges Ableitungsverfahren vorgestellt. Es basiert auf der Wiederverwendung von Signaturen. Das Grundprinzip des Ansatzes besteht in der Ableitung eines Signaturabstraktionsbaums aus einer bereits spezifizierten Signatur. Mit jeder neuen Abstraktionsebene wird die Erkennungsgenauigkeit der Signatur reduziert bis der zu untersuchende Angriff von der Signatur erfasst wird. Auf diese Weise werden dem Signaturmodellierer aus der Menge der bereits spezifizierten Signaturen diejenigen selektiert, die zur Erkennung des neuen Angriffs zur partiellen Wiederverwendung geeignet sind. Anhand

dieser Signaturen kann sich der Modellierer dann bei der Spezifizierung der neuen Signatur orientieren bzw. Entwurfsentscheidungen, sowie Erkennungsmechanismen wiederverwenden. Der beschriebene Ansatz wurde unter Verwendung der Signatursprache EDL [6] bzgl. seiner Anwendbarkeit evaluiert. Allerdings handelt es sich bei EDL um eine Signatursprache zur Beschreibung von *Multi-Step-Attacken*, d. h. Sicherheitsverletzungen zu deren Erkennung mehrere Protokolleinträge in Zusammenhang gebracht werden müssen. In diesem Beitrag wird die Anwendbarkeit des Verfahrens für *Single-Step-Attacken* diskutiert. *Single-Step-Attacken* bezeichnen Sicherheitsverletzungen, die auf Grundlage eines einzelnen Protokolleintrags erkennbar sind. Signaturen zur Erkennung derartiger Attacken beschreiben typischerweise charakteristische Byte-Sequenzen, deren ggf. kombiniertes Auftreten in einem Protokolleintrag auf eine Sicherheitsverletzung hindeutet. Folglich unterscheiden sich Single-Step-Signaturen von Multi-Step-Signaturen nicht nur im Aufbau, sondern auch in den Modellierungsaspekten grundlegend. Als Vertreter für Single-Step-Signaturen betrachten wir die Signatursprache von *Snort* [9].

Der Beitrag ist wie folgt gegliedert: Wir stellen zunächst das in [1] beschriebene Ableitungsprinzip als Grundlage für die Adaption auf die *Snort*-Signatursprache kurz vor. Anschließend beschreiben wir im Abschnitt 3 die Adaption auf die Signatursprache von *Snort*. Abschnitt 4 demonstriert die praktische Anwendung des Ansatzes. Der Beitrag schließt mit einer Zusammenfassung der Ergebnisse und einem Ausblick auf die nächsten Arbeitsschritte.

2 Ableitung von Signaturen durch Wiederverwendung

Der Entwicklungsprozess einer Signatur für eine neue Attacke untergliedert sich in vier Schritte. (1) Zuerst wird der neue Angriff ausgeführt, um die entstehenden Spuren aufzuzeichnen. Spuren sind einzelne sicherheitsrelevante Aktionen, die durch die Sensoren erkannt werden. (2) Der Signaturmodellierer untersucht dann die Spuren und identifiziert die zum Angriff gehörigen. Darauf entwickelt er (3) unter Berücksichtigung der Attackstrategie von Grund auf schrittweise die neue Signatur. Ein wesentlicher Aspekt dabei ist das Erkennen von Mustern, die das Aufdecken von Spuren dieser Attacke durch ein IDS ermöglichen. Nachdem die Signatur spezifiziert ist, wird (4) in der Testphase ihre Korrektheit bzw. Präzision überprüft und die Signatur gegebenenfalls korrigiert. Der Entwicklungsprozess ist aufwendig und fehleranfällig. Eine Wiederverwendung von Signaturen bzw. Signaturfragmenten könnte daher erheblich zur Verkürzung der Entwurfsphase beitragen. Des Weiteren könnte durch die Wiederverwendung von bewährten, d.h. validierten Signaturfragmenten, die aufwendige Test- bzw. Korrekturphase entscheidend verkürzt werden.

Das Prinzip, durch den Einsatz von Mechanismen zur Wiederverwendung von Entwurfs- bzw. Implementierungsentscheidungen aus bereits spezifizierter bzw. entwickelter Software die Entwicklungszeit zu senken, ist aus der Softwaretechnik bekannt. Der Anwendung dieses Prinzips für die Signaturableitung haben sich neben [1] bisher nur wenige Ansätze angenommen. Cheung et al. schlagen vor, den Signaturentwurf durch Verwendung von Angriffsmodellen zu vereinfachen [2]. Dieser Ansatz entspricht den Design Pattern der Softwaretechnik [3]. Die vorgestellten Modelle sind allerdings sehr limitiert

und ermöglichen lediglich eine Wiederverwendung architektureller Entwurfsentscheidungen. Ferner ist eine Wiederverwendung von bereits spezifizierten Signaturen bzw. Signaturfragmenten nicht vorgesehen. Rubin et al. beschreiben einen Ansatz, wie zu einem vorliegenden Angriff Mutanten generiert werden können [4]. Mutanten nutzen die gleichen Schwachstellen wie die Ausgangsattacken, ohne dieselben sicherheitskritischen Aktionen auszuführen. Soll für einen solchen Mutanten eine Signatur entwickelt werden, könnte die Signatur des Ausgangsangriffs, falls vorhanden, zur Wiederverwendung bzw. Weiterentwicklung genutzt werden. Das Verfahren von Rubin et al. könnte durch die Ableitung der abstrahierten Instanz einer Attacke zu diesem Zweck genutzt werden. Allerdings sind die dazu benötigten Transformationsregeln (ausgenommen einfache Transformationen, wie IP-Fragmentierung) stark vom jeweiligen Angriff abhängig. Ein allgemeingültiges Vorgehen für alle Angriffe wäre mit diesem Ansatz nicht realisierbar. Rubin et al. diskutieren in [5] aufbauend auf zwei formalen Sprachen die Weiterentwicklung bzw. Verfeinerung von Signaturen. Dieser Ansatz unterstützt den Signaturmodellierer bei der Beseitigung von Fehlalarmen, die durch unpräzise Signaturen ausgelöst werden. Allerdings setzt dieses Verfahren eine bereits weitgehend fehlerfreie Referenz-Signatur voraus. Larson et al. [8] präsentieren ein Werkzeug, das die Extrahierung von signifikanten Ereignissen, die eine Attacke hinterlässt, aus Auditdaten unterstützt. Dazu wird der Angriff ausgeführt und die entstehenden Auditdaten werden protokolliert. Anschließend wird die Differenz zwischen diesen Auditdaten und einer attackenfreien Audit-Trail ermittelt. Das Problem, aus dieser Differenz eine Signatur abzuleiten, bleibt jedoch offen.

Dem in [1] vorgeschlagenen Ansatz liegt die Annahme zugrunde, dass sich Bestandteile in den Signaturen für die Erkennung ähnlicher Angriffe gleichen. Nach dem Vorschlag von für den neuen Angriff relevanten Signaturen, kann der Signaturmodellierer relevante Parallelen erkennen und für die neue Signatur übernehmen bzw. anpassen. Relevante Signaturen zu einem neuen Angriff werden identifiziert, indem die bekannten Signaturen sukzessive abstrahiert werden bis sie die Spuren des neuen Angriffs erkennen. Dies kann leicht überprüft werden, indem ein IDS die Spuren des Angriffs unter Verwendung der abstrahierten Signatur überprüft. Die Abstraktion der Signaturen erfolgt durch Transformationen. Durch sukzessives Anwenden von Transformationen auf die Ausgangssignatur bzw. deren Abstraktionen lässt sich ein Abstraktionsbaum erzeugen. Jede Transformationsart wird durch eine Metrik gewichtet. Durch die Gewichtung der Transformationen ist es möglich, jeder abstrahierten Signatur ein Ähnlichkeitsmaß in Bezug auf die Ausgangssignatur zuzuordnen. Der Signaturmodellierer kann nun all jene Signaturen näher untersuchen, die am schwächsten abstrahiert werden mussten, um wieder verwendbare Fragmente zu identifizieren. Im Folgenden werden die Transformationen, Abstraktionsbäume, Ähnlichkeitsbemessungen sowie die praktische Realisierung des Ansatzes näher erläutert.

2.1 Transformationen

Eine Signatur spezifiziert eine Menge von Manifestationen eines Angriffs. Unter einer *Manifestation* (Spuren) einer Attacke werden die bei der Durchführung einer Sicherheitsverletzung aufgezeichneten Audit-Datensätze, Ereignisse bzw. Netzwerkpakete ver-

standen. Eine Signatur kann durch das Entfernen bzw. Verändern von semantischen Merkmalen abstrahiert werden. Für solche Transformation muss gelten, dass sich durch deren Anwendung die Menge der Manifestationen, die durch die abstrahierte Signatur erkannt wird, vergrößert. Dementsprechend abstrahiert eine Transformation die Signatur von *S* zu *AS* nur, wenn für die Mengen der von den Signaturen *S* und *AS* erkannten Manifestationen M_S und M_{AS} gilt, dass $M_S \subset M_{AS}$ ist. Die konkreten Ausprägungen der Transformationen sind zwangsläufig stark mit dem verwendeten Signaturmodell gekoppelt. Dementsprechend müssen die Transformationen unter Berücksichtigung auf die konkret verwendete Signatursprache ermittelt werden.

2.2 Abstraktionsbaum einer Signatur

Durch die induktive Anwendung von Transformationen auf eine Ausgangssignatur *S* und den daraus resultierenden abstrahierten Signaturen können alle Abstraktionen von *S* konstruiert werden. Die abstrahierten Signaturen stehen bezüglich ihrer Konstruktion in Relation. Zwei Signaturen *A* und *B* stehen in Relation, falls sich *B* durch Anwendung einer Transformation aus *A* erzeugen lässt. Die Relationen zwischen der Ausgangssignatur *S* und ihren Abstraktionen lassen sich durch einen Baum darstellen.

Signaturen definieren die Menge der erkannten Manifestationen der zugehörigen Attacke. Aufgrund der geforderten Transformationseigenschaft beschreibt die einem Knoten zugeordnete Signatur unter anderem alle Manifestationen der Vatersignatur. Demzufolge beschreibt jede abstrahierte Signatur mindestens die Manifestationen ihrer Vatersignatur. Abb. 1 stellt exemplarisch den Abstraktionsbaum für eine Ausgangssignatur *S* dar. Beispielsweise wird die Abstraktion AS_1 durch Transformation *1* direkt aus *S* erzeugt. Ferner wird die Signatur $AS_{1,7}$ aus AS_1 mittels Transformation *3* erzeugt. Demzufolge gilt für die Manifestationen M_S, M_{AS1}, $M_{AS1,7}$ von den Signaturen *S*, AS_1 und $AS_{1,7}$, *dass* $M_S \subset M_{AS1} \subset M_{AS1,7}$.

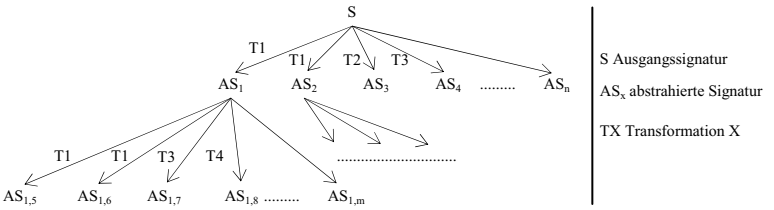

Abb. 1: Abstraktionsbaum und Manifestationsmengen

2.3 Bemessung der Ähnlichkeit von Signaturen

Um die Ähnlichkeit der abgeleiteten Signaturen bewerten zu können, müssen die Kanten des konstruierten Abstraktionsbaums gewichtet werden. Dies erfolgt mittels einer Metrik δ, die jeder Transformation einen Wert zugeordnet. Falls die Signatur *AS* aus der Signatur *S* unter Anwendung der Transformation *X* erzeugt wurde, dann ist die Kante zwischen den zu *S* und *AS* gehörenden Knoten *P* und *P'* mit dem Metrikwert *A* der Transfor-

mation *X* zu gewichten. Nachdem die Kanten im Baum bewertet wurden, kann die Ähnlichkeit unterschiedlicher abstrahierter Signaturen gegenüber der Ausgangssignatur auf der Grundlage der Metrik δ verglichen werden. Dazu werden die Kantengewichte auf dem Pfad vom zur Signatur gehörenden Knoten bis zur Wurzel summiert, diese Summe entspricht dann dem Abstraktionsgrad der Signatur.

2.4 Selektion der Signaturen

Abschließend beschreiben wir das Prinzip der Selektion von wiederverwendbaren Signaturen aus der Menge *K* der gegebenen Signaturen. Die Spuren *T*, die ein neuer Angriff auf einem System hinterlässt, sind aufzuzeichnen. Anschließend wird jeder Ausgangssignatur *S* aus *K* ein Abstraktionsgrad zugeordnet, der angibt wie stark *S* abstrahiert werden muss, damit *S* die Spuren *T* erkennt. Dazu sind die nachfolgenden fünf Schritte für jede Signatur *S* aus *K* auszuführen. (1) Konstruktion aller abstrahierbaren Signaturen, die sich aus *S* mittels der festgelegten Transformationen ableiten lassen. (2) Generierung des zugehörenden Abstraktionsbaumes. (3) Bewertung der Kanten im Baum mittels einer Metrik δ. (4) Analyse der Spuren *T* mit allen aus *S* abgeleiteten Signaturen durch ein IDS. Signaturen, welche die Spuren *T* erkennen, werden markiert. (5) Aus der Menge der markierten Signaturen bzw. Knoten im Abstraktionsbaum wird das geringste Kantengewicht zur Wurzel bestimmt. Dieser Abstraktionsgrad wird für die Signatur *S* vermerkt. Nachdem diese Schritte für alle Signaturen aus *K* durchgeführt wurden, liegt für jede Signatur ein Maß der nötigen Abstraktion vor. Die Signaturen aus *K*, die am wenigsten abstrahiert werden müssen, um die Spuren *T* zu erkennen, werden dem Signaturmodellierer zur Wiederverwendung vorgeschlagen. Anhand dieser Signaturen kann sich der Signaturmodellierer beim Entwurf der neuen Signatur orientieren.

3 Anwendung des Ableitungsprinzips auf *Snort*-Signaturen

Der in Abschnitt 2 beschriebene Ansatz soll nun für die Signatursprache von *Snort* adaptiert werden. *Snort* ist ein netzwerkbasiertes Intrusion Detection System. Es ist mit über 150.000 aktiven Benutzern (Stand: Ende 2006) wahrscheinlich das am häufigsten eingesetzte IDS. Ferner bietet *Snort* eine sehr umfassende und ständig aktualisierte Signaturdatenbank. Wir beginnen mit einer kurzen Übersicht zur Signatursprache von *Snort*, die für das Verständnis der weiteren Ausführungen erforderlich sind. Danach erläutern wir die Anwendung des Transformationsprinzips.

3.1 Signaturbeschreibung mit *Snort*

Die Signaturbeschreibungssprache von *Snort*, eine ausführliche Beschreibung ist in [9] enthalten, erlaubt die Modellierung von Single-Step-Signaturen auf der Basis von Regeln. Die Regeln setzen sich aus einem Regelkopf und einem Optionsteil zusammen. Der *Regelkopf* spezifiziert grundlegende Signaturparameter, wie das zu überwachende Kommunikationsprotokoll (IP, UDP, TCP, ICMP), die zu überwachende Kommunikationsrichtung (bi-/uni-direktional), sowie die Quell- bzw. Zieladressen/Portnummern. Außer-

dem kann festgelegt werden, welche Aktion bei einem erkannten Angriff auszulösen ist. Die Möglichkeiten reichen hierbei von der Generierung von Alarmen (*msg*) bis zum Verwerfen (*drop*) von Paketen. Im *Optionsteil* werden Filterbedingungen für den Payload- und den Nicht-Payload-Bereich der Pakete festgelegt. Die Payload-Optionen beziehen sich hierbei direkt auf die Nutzdaten der zu untersuchenden Pakete. Mittels dieser Optionen kann nach Mustern bzw. konkreten Werten in den Nutzdaten der Pakete bzw. Bytesträme gesucht werden. Dagegen beschreiben die Nicht-Payload-Optionen Bedingungen bzgl. der Header der Protokolldateneinheiten. Die Optionen sind jeweils abhängig von dem im Regelkopf spezifizierten zu überwachendem Protokoll.

Wird durch *Snort* ein Paket oder ein Bytestrom identifiziert, der sämtliche spezifizierte Bedingungen erfüllt, so wird eine Aktion ausgelöst. Das konkrete Verhalten der Aktion wird durch Post-Detection- und Metadaten-Optionen im Optionsteil der Regel festgelegt. Da die Aktionen nicht im Zusammenhang mit dem Analyseprozess der Audit-Daten stehen, werden diese nicht näher erläutert. Abb. 2 zeigt ein Beispiel einer *Snort*-Signatur mit Kennzeichnung des Regelkopfes, der Filterbedingungen für Payload und Nicht-Payload-Bereiche. Die dabei verwendeten konkreten Snort-Optionen werden im nächsten Abschnitt im Zusammenhang mit den Abstrahierungsmöglichkeiten erläutert.

```
alert tcp any any  -> 192.168.1.0/24 445 (      //Regelkopf
       tos: 1;                                  //Nicht-Payload-Optionen
       flow: to_server, established;
       content: "|FF|SMB%";depth:5; offset:4;   //Payload-Optionen
       content: "&|00|"; within:2; distance:56;
       content: "|05|"; within:1; distance:2;
       content: "|0B|"; within:1; distance:1;
       byte_test:1,&,1 ,0, relative;
       content: "|00|" ;within: 1; distance:21 ;
       msg: " Netbios access" ;                  //Metadaten-Optionen
       sid: 2191;)
```

Abb. 2: Beispiel einer *Snort*-Signatur

3.2 Abstraktion von *Snort*-Signaturen

Wir beschreiben in diesem Abschnitt die Abstraktion von *Snort*-Signaturen. Dazu werden acht grundlegende Transformationsregeln vorgestellt. Die Transformationen können in drei Klassen unterteilt werden: *Abstrahierung der grundlegenden Kommunikationsparameter des Regelkopfes*, der *Optionen des Protokolldaten-Headers* (IP bzw. TCP) und der *Payload-Optionen*. Die Transformationen werden nachfolgend näher vorgestellt. Dazu werden die jeweilig abstrahierte *Snort*-Option, die Realisierung der Transformation und, falls notwendig, Anmerkungen erläutert. Eine Transformation abstrahiert eine Signatur S entsprechend des obigen Ansatzes nur, wenn für die resultierende Signatur AS, die Ausgangssignatur S und die zugehörigen Mengen M_S bzw. M_{AS} der Manifestation gilt, dass $M_S \subset M_{AS}$ ist. Dies ist nicht bei allen Transformationen zwangsläufig gegeben. Die Zuordnung von Vorbedingungen stellt diesen Sachverhalt sicher. Die folgenden Transformationen und Erläuterungen beschränken sich aufgrund des begrenzten Umfangs dieses Beitrags auf die wichtigsten *Snort*-Optionen. Eine vollständige Beschreibung aller Transformationen ist in [10] zu finden.

3.2.1 Abstrahierung der Kommunikationsparameter des Regelkopfes

Im Regelkopf einer *Snort*-Signatur kann der zu überwachende Netzverkehr eingeschränkt werden. Dazu können konkrete Quell- bzw. Ziel-IP-Adressen bzw. Netzwerksegmente spezifiziert werden. Ferner kann der zu prüfende Verkehr durch Quell und Ziel-Portnummern eingeschränkt werden. Nur Netzverkehr der zwischen den spezifizierten Kommunikationspunkten fließt, wird dann bzgl. der Signatur analysiert. Werden diese Einschränkungen aufgehoben, so wird der gesamte protokollierte Verkehr analysiert.

Realisierung: Einschränkende Angaben zu IP-Adressen, Netzwerksegmenten oder Portnummern werden durch das Schlüsselwort *any* im Regelkopf der Signatur ersetzt.

Anmerkungen: Existieren keine weiteren Payload- bzw. Nicht-Payload-Optionen, so deckt die Signatur lediglich beliebigen Verkehr zwischen den spezifizierten IP-Adressen bzw. Ports auf. Ist das der Fall sollte die Signatur vom Abstraktionsprozess ausgeschlossen werden, da eine spätere Wiederverwendung nicht sinnvoll ist.

3.2.2 Abstrahierung der Optionen des IP-Protokolldaten-Headers

Im Optionsteil einer *Snort*-Signatur können mittels IP-Optionen Bedingungen über den relevanten IP-Header-Feldern der zu überwachenden Pakete definiert werden. Das betrifft die Optionen für die IP-Headereinträge *TTL*, *TOS*, *ID*, *Fragbits*, *Fragoffset*, *Paketgröße*, *Kommunikationsprotokoll* sowie die *optionalen IP-Optionen*. Beispielsweise lässt sich durch die *Fragmentbits*-Option die Analyse auf fragmentierte Pakete beschränken. Fordert eine Snort-Signatur die Belegung von IP-Feldern mit bestimmten Werten, so werden alle Pakete, die diese Bedingungen nicht erfüllen, von der weiteren Analyse ausgeschlossen. Das Vorkommen von weiteren in der Signatur beschriebenen Mustern (z.B. bestimmte Bytestrings im Payload) wird so nicht geprüft. Durch das Entfernen von den genannten IP-Optionen werden auch alle anderen IP-Pakete weiter analysiert.

Realisierung: Einfache einschränkende IP-Optionen, wie *TTL*, *TOS*, *ID*, Fragbits, Fragoffset, Paketgröße oder Einschränkungen bzgl. des transportierten Kommunikationsprotokolls, werden einzeln pro Transformation aus der Regel entfernt.

3.2.3 Abstrahierung der Optionen des TCP-Protokolldaten-Headers

Analog zu den IP-Optionen lassen sich mittels der *Snort*-TCP-Optionen Bedingungen über zu untersuchende TCP-Segmente spezifizieren. Diese Bedingungen beziehen sich dabei auf die Felder des Segment-Headers. In *Snort*-Signaturen können folgende TCP-Optionen spezifiziert werden. Mittels der Option *flags* können die einzelnen Kontrollbits wie *FIN*, *SYN* usw. getestet werden. Die Option *flow* realisiert die Einschränkung der Analyse auf offene, verbundene oder zustandslose TCP-Verbindungen und durch die zusätzliche Option *from_client* oder *from_server* auf Verkehr in entsprechender Kommunikationsrichtung. Ferner kann bspw. mit der Optionen *seq*, *ack* und *window* die Analyse auf Segmente mit bestimmten Sequenznummern bzw. Bestätigungen oder auf Segmente mit dedizierten Fenstergrößen begrenzt werden. Alle von *Snort* unterstützten TCP-Optionen implizieren wie die IP-Optionen Einschränkungen der Analyse auf bestimmte Pakete. Werden diese Einschränkungen aufgehoben, werden alle Pakete auf die restlichen Muster der *Snort*-Signatur untersucht.

Realisierung: Einschränkende TCP-Optionen werden einzeln pro Transformation aus der *Snort*-Signatur entfernt.

3.2.4 Abstrahierung der Payload-Optionen

Zur Spezifikation von Mustern, nach denen die Nutzdaten der Pakete analysiert werden sollen, werden in *Snort* Payload-Optionen genutzt. Sie erlauben die Suche nach charakteristischen Spuren eines Angriffs in den Nutzdaten von IP-Paketen oder TCP-Bytetströmen, somit lassen sich bspw. Protokollnachrichten der Anwendungsschicht auf bestimmte Muster untersuchen. Die Payload-Optionen und deren Transformationen sehen im Einzelnen wie folgt aus.

content-Option: Diese Option fordert das Vorkommen der spezifizierten Zeichenkette in den Nutzdaten eines IP-Pakets (oder eines TCP-Bytestroms) und wird genutzt, um Protokollnachrichten oder Steuerkommandos höherer Protokolle zu erkennen, die versuchen eine Verwundbarkeit auszunutzen.

Realisierung: Content-Optionen lassen sich durch drei Transformationen abstrahieren. (1) Das *Content*-Schlüsselwort wird um eine *nocase*-Option ergänzt. Dies führt zur Abstrahierung der Signatur, da der spezifizierte String auch *case insensitive* in den Nutzdaten auftreten darf. (2) Enthält die gesuchte Zeichenkette Trennzeichen (z.B. Leerzeichen, Tabs usw.), dann wird die ursprüngliche Zeichenkette an den Trennzeichen unterteilt und es wird jeweils für jeden Teilstring ein neues *content*-Schlüsselwort eingeführt. Die signifikanten Teile des gesuchten Strings müssen somit nicht mehr zwingend in den ursprünglich definierten Abständen auftreten bzw. von den definierten Trennzeichen eingeschlossen sein. Modifikationen von Angriffen, bei denen bspw. lediglich die Anzahl von Leerzeichen variiert, werden durch diese Abstraktion erkannt. (c) Als letzte Abstraktionsmöglichkeit wird die *content*-Option verworfen.

Vorbedingungen: Bei Signaturen mit mehren Payload-Optionen können diese Optionen untereinander in Beziehung gesetzt werden. Dies erlaubt die Spezifikation der Reihenfolge des Auftretens bzw. bestimmten Abständen zwischen den einzelnen Optionen. Eine *content*-Option kann somit nur verworfen werden, wenn keine weiteren Optionen auf sie Bezug nehmen. Andernfalls sind zuerst bezugnehmenden relativen Suboptionen *offset, depth, distance* und *within* zu entfernen.

uricontent-Option: Diese Option durchsucht den Inhalt eines Paketes nach der spezifizierten, normalisierten URI-Zeichenkette. Normalisiert bedeutet dabei, dass alle semantisch identischen Zeichenketten erkannt werden, auch wenn sie besonders codiert wurden. Zusätzlich werden durch *Snort* bei der Verwendung dieser Option Verzeichnispfad-Anweisungen erkannt. Enthält bspw. ein TCP-Bytestrom im Nutzdatenteil den Pfad <sub-dir>/../../< neighbor-dir>, wird der Pfad auf ../<neighbor-dir> normalisiert.

Realisierung: Die *uricontent*-Option wird verworfen.

Anmerkungen: Sinnvolle Möglichkeiten zur Manipulation von URI-Zeichenketten oder Pfadangaben sind stark von der jeweiligen Angriffsstrategie bzw. Verwundbarkeit ab-

hängig, so dass als generelle Abstraktionsmöglichkeit lediglich das Verwerfen der ganzen Option in Frage kommt.

isdatat-Option: Mit dieser Option wird überprüft, ob Pakte oder TCP-Byteströme Daten an der in der Option festgelegten Position enthalten. Diese *isdatat*-Option wird in *Snort*-Signaturen hauptsächlich zum Aufdecken von Bufferoverflows genutzt.

Realisierung: Die *isdatat*-Option wird verworfen.

Anmerkungen: Analog zur *uricontent*-Option ist eine allgemeine sinnvolle Abstraktion dieser Option nicht möglich, da diese stark von der jeweiligen Verwundbarkeit abhängig ist. Dementsprechend bleibt auch hier nur das Verwerfen der ganzen Option als Transformationsmöglichkeit.

pcre-Option: Ähnlich der *content*-Option kann mittels der *pcre*-Option das Vorkommen von speziellen Zeichenketten oder Mustern in den Nutzdaten von IP-Paketen oder TCP-Bytteströmen geprüft werden. Bei der *pcre*-Option werden die Muster allerdings mittels Perl-kompatibler regulärer Ausdrücke spezifiziert. Dies erlaubt erheblich größere Spezifikationsfreiheiten als bei der *content*-Option. Ferner können durch pcre-Optionsschalter die Interpretation des regulären Ausdrucks und das Matching beeinflusst werden.

Realisierung: Eine *pcre*-Option kann zur Abstraktion einer *Snort*-Signatur abgeschwächt oder verworfen werden. Die Abschwächung kann auf zwei Arten realisiert werden: a) Optionsschalter werden verändert, so dass die Analyse des regulären Ausdrucks ausgeweitet wird. So kann bspw. der Ausdruck unabhängig von Groß- oder Kleinschreibung analysiert werden. Als zweite Transformationsart wird b) der reguläre Ausdruck an sich transformiert. Hier werden bspw. vorhandene Zeilenumbrüche wie '\r\n', '\n\r', '\r' oder '\n' durch '\R' ersetzt, so dass alle Kodierungen von Zeilenumbrüchen erfasst werden. Ferner können *'^'* und *'$'*-Zeichen an Anfang/Ende des Ausdrucks entfernt werden. Der resultierende Ausdruck ist dann zusätzlich gültig, wenn er Daten erfasst, die sich nicht am Anfang bzw. Ende des Puffers oder einer Zeile befinden. Weiterhin können im Ausdruck enthaltene '\s' durch '\s+' ersetzt werden. Somit erkennt der resultierende Ausdruck auch Zeichenketten, in denen die Anzahl der Leerzeichen variiert. Die letzte Abstraktionsmöglichkeit ist das Teilen des regulären Ausdrucks an Trennzeichen (z.B. Leerzeichen). Hierbei wird die ursprüngliche pcre-Option durch zwei Optionen ersetzt.

Vorbedingungen: pcre-Optionen können mit anderen Payload-Optionen in Beziehung stehen. Demzufolge müssen vor dem Verwerfen der Option eventuell bezugnehmende Suboptionen entfernt werden.

byte_jump-Option: Diese Option extrahiert eine vorgegebene Anzahl von Bytes an definierter Stelle aus dem Datenstrom und interpretiert diese als Sprungadresse zu einer anderen Position im Datenstrom. Anschließend können andere Payload-Optionen relativ zur gefundenen Stelle ausgewertet werden.

Realisierung: Da es sich bei einer *byte_jump*-Option um eine sehr angriffsspezifische Überprüfung handelt, für die keine generelle Abstraktion möglich ist, kann diese Option in einer *Snort*-Signatur nur abstrahiert werden, indem sie komplett verworfen wird.

Vorbedingungen: Die zu entfernende *byte_jump*-Option darf nicht in Bezug mit anderen Optionen stehen. Ist dies der Fall, muss zuvor diese Abhängigkeit aufgelöst werden.

Das komplette Verwerfen einer Option als einzige Abstraktionsmöglichkeit von Optionen (z.B. bei *uricontent*-, *isdatat*- und *byte_jump*) scheint auf den ersten Blick nicht sinnvoll, allerdings muss dabei beachtet werden, dass a) *Snort*-Signaturen im Schnitt aus sieben unterschiedlichen Optionen bestehen (bis hin zu 22 verschieden Optionen) und b) eine Option meist nur eine typische Ausprägung oder Eigenheit des zu erkennenden Angriffs erfasst. Somit verbleiben auch beim Verwerfen von mehren Optionen noch genügend semantische Inhalte, die einen Test der Signatur im Zusammenhang mit den Spuren des neuen Angriffs rechtfertigen.

4 Anwendungsbeispiel

Um die Eignung des Verfahren detailliert zu evaluieren, müsste ein Kreuztest von mehreren *Snort*-Signatur-Entwicklungszyklen mehrerer „erfahrener" Signaturentwickler mit und ohne dem vorgestellte Verfahren durchgeführt werden. Durch den Vergleich der relevanten Prozessparameter, z.B. die Entwicklungszeiten sowie die Qualität der erstellten Signaturen, könnte dann die Eignung des Verfahrens nachgewiesen werden. Ein solches Vorgehen würde aber einen sehr hohen personellen und finanziellen Aufwand erfordern. Um trotzdem einen Eindruck der Eignung des vorgestellten Verfahrens zu erhalten, wurde das gesamte Abstrahierungs- und Selektionsverfahren für Snort-Signaturen in einem Werkzeug prototypisch umgesetzt und mehrfach exemplarisch evaluiert. Dabei übernimmt das implementierte Werkzeug automatisiert alle Schritte, von der Abstrahierung der Signaturen, über den Test der abstrahierten Signaturen im Zusammenhang mit Snort und den Spuren des neuen Angriffs, bis hin zur Selektion der wiederverwendbarer Signaturen. Das Werkzeug wurde mehrfach exemplarisch evaluiert und die dabei gewonnen Ergebnisse sind durchweg positiv. Bevor nun ein konkretes Anwendungsbeispiel beschrieben wird, wird die allgemeine Vorgehensweise der Evaluierung erläutert.

Die Grundlage für die Menge der gegebenen wiederverwendbaren Signaturen bilden die Snort-VRT *Certified Rules* vom 14.05.2007. Diese Regelmenge besteht aus 7544 *Snort*-Signaturen mit durchschnittlich sieben Optionen pro Regel. Die Evaluierung des Verfahrens erfolgte immer in drei Phasen. (a) Eine beliebige Signatur wird aus der Signaturdatenbank entfernt. (b) Die Spuren des zu der entfernten Signatur gehörenden Angriffs werden erzeugt bzw. synthetisiert und aufgezeichnet. (c) Das Werkzeug transformiert die in der Signaturdatenbank verbliebenden *Snort*-Signaturen gemäß dem vorgestellten Ableitungsprinzip und generiert alle abstrahierbaren Signaturen. Die generierten Signaturen werden anschließend automatisiert gegen die aufgezeichneten Spuren des Angriffes getestet. d) Als letzten Schritt listet das Werkzeug die abstrahierten Signaturen, die den Angriff erkannt haben, nach ihrem Abstraktionsgrad auf. Zur Bemessung der Ähnlichkeit wird eine Metrik δ verwendet, die jeder Transformation den Wert 1 zuordnet.

Um einen Eindruck der Eignung des Verfahrens zu erhalten, wird nun ein konkreter Evaluierungszyklus beschrieben. Als Anwendungsfall wurde eine Signatur ausgewählt, die einen Bufferoverflow-Angriff auf den FTP-Dienst der Oracle-XML-Datenbank erkennt.

Die Signatur ist in Abb. 3 dargestellt. Sie zeichnet sich dadurch aus, dass sie einen TCP-Bytestrom vom Client zum Server auf Port 2100 erkennt, der in den Nutzdaten einen überlangen String mit „USER“, gefolgt von einem oder mehreren Leerzeichen und beliebigen anderen Zeichen und einer Gesamt-String-Länge von über 100 Zeichen enthält. Der zugehörige Angriff versucht, während der Einlog-Prozedur auf einen Oracle-FTP-Dienst durch eine überlange Nutzerkennungsnachricht einen Bufferoverflow auszulösen.

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 2100 (flow: to_server,
       established; content: " user"; nocase;
       pcre: "/^USER\s+[^\n]{100,}/smi; sid: 3631;)
```

Abb. 3: *Snort*-Signatur für einen Oracle-FTP Pufferüberlauf

In Abb. 4 sind zeilenweise die vier Signaturen mit dem geringsten Abstraktionsgrad dargestellt, d.h. die Signaturen, die am wenigsten abstrahiert werden mussten, um die Spuren des Oracle-FTP-Angriffs zu erkennen. Die jeweils zu transformierenden Optionen sind dabei grau hinterlegt.

Abstraktionsgrad / abstrahierte Signatur	
3	`alert tcp any any -> any 21 (flow:established, from_client; content:"USER"; nocase; pcre:"/^USER\s[^\n]*?%[^\n]*?%/ism"; sid:2178;)`
4	`alert tcp any any -> any 3535 (flow:established, from_client; content:"USER"; nocase; content:"y049575046"; nocase; pcre:"/^USER\s+y049575046/ism";sid:2334;)`
4	`alert tcp any any -> any 21 ( flow : established, from_client; content: "USER" ; nocase ; isdataat : 100 ,relative; pcre:"/^USER(?!\n)\s[^\n]{100}/ism" ; sid : 1734; )`
4	`alert tcp any any -> any 21 ( flow : established, from_client; content: "USER" ; nocase; content: "w0rm"; nocase; distance:1; pcre :"/^USER\s+w0rm/ism" ; sid: 144; )`

Abb. 4: *Signaturen mit dem geringsten Abstraktionsgrad*

Die erste Signatur erkennt den Oracle-FTP-Angriff, wenn die Einschränkung auf Port 21 aufgehoben wird, da der Angriff auf Port 2100 erfolgt. Ferner muss die *pcre*-Option am Leerzeichen in zwei *pcre*-Optionen getrennt werden und anschließend die entstandene zweite *pcre*-Option verworfen werden. Die durch diese Transformationen entstandene abstrahierte Signatur hat einen Abstraktionsgrad von drei. Sie musste von allen Signaturen aus der Datenbank am wenigsten abstrahiert werden, um die Spuren des Oracle-FTP-Angriffes zu erkennen. Die zweite vorgeschlagene Signatur besitzt einen Abstraktionsgrad von vier. Bei dieser Signatur musste ebenfalls die Porteinschränkung aufgehoben und die *pcre*-Option geteilt und die dadurch entstandene zweite *pcre*-Option verworfen werden. Weiterhin musste die *content*-Option, die das Auftreten des Strings „y049575046“ fordert, verworfen werden. Die dritte Signatur muss ebenfalls durch vier Transformationen abstrahiert werden. Hier mussten die Porteinschränkung aufgehoben und die *isdatat*-Option verworfen werden. Ferner musste der „\s“-Teil der *pcre*-Option, der das Auftreten eines Leerzeichens fordert, in „\s+“ transformiert werden. Bei der vierten am wenigsten zu abstrahierenden Signatur musste die Porteinschränkung aufgehoben werden. Weiterhin wurde die *content*-Option verworfen und ebenfalls die *pcre*-Option getrennt, wobei die entstandene zweite *pcre*-Option ebenfalls verworfen wurde.

Als Signatur zur Wiederverwendung für den Oracle-FTP-Angriff sollte die dritte vorgeschlagene Signatur verwendet werden. Um den Angriff zu erkennen, kann fast eins zu eins die *pcre*-Option übernommen werden. Lediglich eine geringe Anpassung, „\s+“ statt „\s“, und das Entfernen von „(?!\n)“ sind notwendig. Die eigentliche Signaturcharakteristik, das Erkennen von überlangen Strings mit führendem „user“ gefolgt von Leerzeichen, bleibt dabei erhalten. An diesem Beispiel ist erkennbar, dass der vorgeschlagene Ansatz zur Wiederverwendung von Signaturen auf *Snort*-Signaturen adaptierbar ist und gute Erfolge zeigt. Jedoch ist während der experimentellen Evaluierung auch gleichzeitig eine Schwäche der prototypischen Implementierung sichtbar geworden: die verwendete sehr einfache Metrik, die zur Bemessung der Abstraktion verwendet wurde, ist in manchen Fällen suboptimal, d.h. die Reihenfolge der vorgeschlagenen Signaturen entspricht nicht immer der Eignung zur Wiederverwendung. Bei der experimentellen Evaluierung traten Fälle auf, bei denen sich die am besten zur Wiederverwendung geeignete Signatur nicht unter den zehn am wenigsten zu abstrahierenden Signaturen befand, sondern erst im Bereich 10 bis 25. Dieses Problem lässt sich durch eine verfeinerte Metrik beheben.

5 Zusammenfassung und Ausblick

Durch die Wiederverwendung von Signaturentwurfsentscheidungen bzw. Fragmenten aus bereits spezifizierten Signaturen kann der Entwurfsprozess von neuen Signaturen unterstützt werden. Wiederverwendung von bewährten Strukturen kann nicht nur den Aufwand für den Entwurfsprozess der neuen Signatur reduzieren, sondern auch die aufwendige Test- bzw. Korrekturphase kann möglicherweise entscheidend verkürzt werden. Dieser Beitrag zeigt, wie sich das in [1] vorgestellte allgemeine Verfahren auf die Signatursprache von Snort adaptieren lässt. Dazu wurden die entscheidenden Elemente der Signaturbeschreibung von Snort analysiert und geeignete Transformationen zur Abstrahierung von Snort-Signaturen identifiziert. Ferner wurde der gesamte Abstraktions- und Selektionsprozess für Snort in einem Werkzeug umgesetzt. Das Werkzeug selektiert automatisch zu den Spuren eines neuen Angriffs wiederverwendbare Signaturen aus einer gegebenen Signaturdatenbank. Die identifizierten Transformationen und der Ansatz wurde im Zusammenhang mit Snort mittels des Werkzeugs mehrfach exemplarisch erprobt und es wurden durchweg positive Resultate verzeichnet. Somit ist das allgemeine Verfahren aus [1] nicht auf Multi-Step-Signaturen begrenzt, sondern lässt sich auch auf Single-Step-Signaturen anwenden.

In weiteren Arbeiten sollen anspruchsvollere Ähnlichkeitsmetriken im Zusammenhang mit den identifizierten Snort-Transformationen erprobt werden. Ferner soll der Abstraktionsprozess so modifiziert werden, dass sich wiederkehrende Muster in der Snort-Signaturdatenbank identifizieren lassen. Diese wiederkehrenden Strukturen könnten dann ggf. zu Entwurfsmustern verallgemeinert werden, so dass analog zu den Entwurfsmustern in objektorientierter Software (s. [3]) der Signaturmodellierer durch die Wiederverwendung von bewährten Strukturen/ Konstrukten bei der Spezifikation von neuen Signaturen unterstützt wird.

6 Literatur

[1] Schmerl S.; König H.; Flegel U.; Meier, M.: Simplifying Signature Engineering by Reuse. In Müller, G. (ed.): Proceedings of the International Conference on Emerging Trends in Information and Communication Security (ETRICS 2006), LNCS 3995, pages 436-450 , Freiburg, Germany, June 2006, Springer , ISBN 3-540-34640-6.

[2] Cheung S.; Lindqvist U.; Fong M.: Modeling Multistep Cyber Attacks for Scenario Recognition. In Proceedings of the 3rd DARPA Information Survivability Conference and Exposition, Washington, USA, IEEE Computer Society Press, pp. 284-292, 2003.

[3] Gamma E., Helm R., Johnson E. R., "Design Patterns – Elements of Reusable Object-Oriented Software", Addison-Wesley Professional, 1997.

[4] Rubin S., Jha S., Miller B.: Automatic Generation and Analysis of NIDS Attacks. In Proceedings of the 20th Annual Computer Security Applications Conference, Tucson, AZ, USA, IEEE Computer Society Press, pp. 28-38, 2004.

[5] Rubin S.; Jha S.; Miller P. B.: Language-based generation and evaluation of NIDS signatures. In Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA, USA, IEEE Computer Society Press, pp. 3-17, 2005.

[6] Meier M.; Schmerl S.: Improving the Efficiency of Misuse Detection. In: Proceedings of the Second Conference on "Detection of Intrusions & Malware and Vulnerability Assessment" (DIMVA2005), Vienna, Austria, July 7-8 2005, LNCS 3548, pp. 188 - 205, Springer 2005.

[7] Lippmann, R.; Webster, S.; Stetson, D.: The Effect of Identifying Vulnerabilities and Patching Software on the Utility of Network Intrusion Detection. In Proceedings of the Symposium on Recent Advances in Intrusion Detection, Zurich, Switzerland, Lecture Notes in Computer Science, Vol. 2516, pp. 307-326, 2002.

[8] Larson U., Lundin Barse E., Jonsson E.: METAL - A Tool for Extracting Attack Manifestations. In: Proceedings of the Second Conference on "Detection of Intrusions & Malware and Vulnerability Assessment" (DIMVA2005), Vienna, Austria, July 7-8 2005, LNCS 3548, pp. 85-102, Springer 2005.

[9] Baker, A.: Beale J.; Caswell B.; Poore M.: Snort 2.1 Intrusion Detection. Syngress Publishing, 2004.

[10] Rietz, R.: Vereinfachung der Signaturentwicklung durch Wiederverwendung. Bachelorarbeit, Lehrstuhl Rechnernetze und Kommunikationssysteme, BTU-Cottbus, 2007.