# Online-Erkennung von IDS-Ereigniskombination mittels XSLT*

Frank Außerlechner
Fachhochschule Koblenz
FB Elektrotechnik und Informationstechnik
Konrad-Zuse-Straße 1
56075 Koblenz
frank@ausserlechner.de

Marko Jahnke, Michael Bussmann, Sven Henkel
Forschungsgesellschaft für Angewandte Naturwissenschaften e.V. (FGAN)
Institut für Kommunikation, Informationsverarbeitung und Ergonomie (FKIE)
Neuenahrer Straße 20
53347 Wachtberg
{jahnke,bus,henkel}@fgan.de

## 1 Einführung

Die Erkennung von miteinander korrelierten Ereignismeldungen gehört im Bereich der Intrusion-Detection seit mehreren Jahren zu den Forschungsgebieten, denen eine erhöhte Aufmerksamkeit geschenkt wird. Ereignismeldungen (oder *Events*) sind Meldungen über potentiell sicherheitsrelevante Ereignisse in Computersystemen oder -netzwerken und werden von Sicherheitswerkzeugen (Intrusion-Detection-Systeme, Firewalls/Paketfilter, Virenscanner, Integritätsprüfprogramme etc.) erzeugt.

Die Disziplin der *Event Correlation* bzw. *Fusion* hat sich zum Ziel gesetzt, bestimmte Ereigniskombinationen (d.h. Mengen miteinander korrelierter Ereignismeldungen), die in ihrer Gesamtheit beispielsweise mit dem typischen Vorgehen eines Angreifers übereinstimmen (*Attack Sequence*) zu erkennen und geeignete Gegenmaßnahmen einzuleiten. Der Mehrwert solcher Verfahren ist zunächst in der gesteigerten Erkennungsleistung zu sehen, da mitunter völlig harmlos erscheinende Einzelmeldungen unter bestimmten Voraussetzungen ein Zeichen für eine schwerwiegende Bedrohung gegen Rechnersysteme oder -netze sein können.

Die meisten bekannten Verfahren zur Erkennung solcher Meldungskombinationen arbeiten unter Verwendung von Datenbanken der erfassten Meldungen und fallen daher in die Kategorie *Offline-Verfahren*. Diese Verfahren können zu jedem Zeitpunkt zeitlich zurückliegende Kombinationen im Meldungsbestand aufdecken. Zur Generierung der

1 * Veröffentlicht in: Tagungsband zum 12. DFN-CERT/PCA-Workshop „Sicherheit in verteilten Systemen“, Hamburg, März 2005.

entsprechenden Korrelationsregeln, die diese Zusammenhänge beschreiben, werden oftmals statistische Auswertungen dieser Datenbanken, aber auch Modelle des betrachteten Zielnetzwerkes, seiner Komponenten und seiner Verwundbarkeiten herangezogen.

Dieser Beitrag beschäftigt sich im Kontrast dazu mit einem *Online-Erkennungsverfahren*, das direkt auf dem Strom der erfassten Ereignismeldungen arbeitet und zuvor spezifizierte Ereigniskombinationen (wieder-) erkennen kann. Zwar sind durch Begrenzung des Speicherbedarfs nur begrenzt zeitlich zurückliegende Meldungen miteinziehbar; jedoch besteht die Möglichkeit, diese Verfahren an beliebigen Stellen des Meldungsstroms verteilt einzusetzen.

Bei der Verarbeitung von heterogenen Mengen von Ereignismeldungen geht man sinnvoller Weise von der Verwendung eines (Quasi-)Standards für das Datenmodell und das Austauschformat von Ereignismeldungen aus. Es handelt sich hier um das Intrusion Detection Message Exchange Format (IDMEF) der IETF Intrusion Detection Working Group (IDWG), die in Form einer XML-DTD (eXtensible Markup Language Document Type Definition) spezifiziert sind [1]. Für die Verarbeitung von XML-formatierten Ereignismeldungen bieten sich daher Verfahren wie XSLT (eXtensible Stylesheet Language Transformations [2]) an, die beispielsweise bei der Umsetzung von XML-Quellen in verschiedene Dokumentenformate (HTML, PDF etc.) Verwendung finden.

# 2 Bedingte Musterübereinstimmungsprüfung und -transformation mittels XSLT

Mittels XSLT lässt sich eine bedingte Musterübereinstimmungsprüfung und -transformation (Conditional Pattern Matching and Transformation, CPMT) für Ereignismeldungen sehr flexibel realisieren. Grundlage für jeden CPMT-Transformationsvorgang ist die Spezifikation einer *Transformations-Regel*, die sich aus *Matching-Templates* und *Transformations-Templates* zusammensetzen. Matching-Templates spezifizieren die Mengen von Ereignismeldungen, die einer Transformation unterworfen werden sollen, und Transformations-Templates beschreiben deren Resultat (eine formale Beschreibung der Technik findet sich in [4] und [5]).

Die einfachste Anwendung des Verfahrens ist eine konstante 1:1-Transformation von Ereignismeldungen, die (nicht-kontextsensitiv) ineinander überführt werden. Die Anwendungen dafür sind offensichtlich begrenzt. Als erste und einfachste Anwendung von CPMT stellte sich die Informationsbereinigung bei Ereignismeldungen dar.

## 2.1 Beispielproblem Informationsbereinigung

In verschiedenen Konstellationen ist es angebracht, Informationen über erkannte Angriffe auf Rechner und Rechnernetze zwischen verschiedenen Domänen auszutauschen (Beispiel: Koalitionsumgebungen). Dieser Austausch von Ereignismeldungen kann u. U. zu einer gesteigerten Möglichkeit führen, synchronisierte und großflächig verteilte Angriffe zu erkennen. Bevor Ereignismeldungen allerdings eine Domäne verlassen, müssen Sie entsprechend der lokalen Richtlinie zur Informationsweitergabe (*Information Sharing Policy, IShP*) bereinigt (d.h. anonymisiert bzw. pseudonymisiert) werden, damit keine sensiblen Informationen weitergeleitet werden. Dies kann beispielsweise durch entsprechende Gateways an den Domänengrenzen bewerkstelligt werden (vgl. [3]).

Unglücklicherweise kann ein sinnvoller Informationsbereinigungsprozess nicht ausschließlich auf einer statischen Textsubstitution basieren. Wenn beispielsweise IP-Adressen als Bestandteil von Meldungen durch feste Werte ersetzt werden sollen, gehen alle Informationen über die Topologie des Netzwerks verloren. Offensichtlich behindert dies den Erkennungsprozess, insbesondere, wenn Verkehrsbezogene Anomalien entdeckt werden sollen. Daher benötigt man eine flexible Methode, um Transformationsregeln zu spezifizieren. Weiterhin kann es erforderlich sein, dass eine Normalisierung von Ereignismeldungen realisiert werden muss (z. B. wenn dasselbe Ereignis mit unterschiedlichen Signaturbezeichnungen oder Referenzen auf Angriffsdatenbanken gemeldet wird).

Dieses Problem kann durch die Verwendung von XSLT gelöst werden. Grundlage ist ein erweiterter XSL-Prozessor, der zusätzlich Reguläre Ausdrücke (POSIX.1-REs) sowie boolesche und arithmetische Ausdrücke verarbeiten kann. In dem folgenden Beispiel wird ein kombiniertes Matching- und Transformations-Template als Teil einer modifizierten IDMEF-Nachrichtensyntax spezifiziert:

```
(1)  <IDMEF-Message xmlns:xsl=...>
(2)    <Alert>...
(3)      <address>
(4)        192\.22\.([0-9]{1,3})$X$\.([0-9]{1,3})$Y$
(5)        <condition>
(6)         ($Y<255)
(7)        </condition>
(8)        <transform>
(9)          <xsl:copy>
(10)           191.72
(11)              .<xsl:value-of select="$X"/>
(12)              .<xsl:value-of select="$Y"/>
(13)         </xsl:copy>
(14)       </transform>
(15)     </address>...
(16)   </Alert>
(17) </IDMEF-Message>
```

Die ersten beiden Bytes einer IP-Adresse `192.22.x.y` sind fest vorgegeben, während die letzten beiden Bytes durch reguläre Ausdrücke dargestellt werden (Zeile (4)). Gleichzeitig werden sie durch Klammerung qualifiziert (sog. *Submatchings*) und der bei einer Übereinstimmungsprüfung auf sie zutreffende Text Variablen `$X` und `$Y` zugewiesen, die man in anderen Bereichen der Spezifikation referenzieren kann. In diesem Falle werden die Inhalte der Submatchings in die resultierende Meldung an der angegebenen Stelle wieder eingesetzt (Zeilen (10)-(12)), so dass sich eine Ersetzung des IP-Präfixes `192.22.` nach `191.72.` ergibt. Zusätzlich kann man arithmetisch-boolsche Bedingungen für die Durchführung der Transformation formulieren. Zeilen (5)-(7) legen beispielsweise fest, dass das letzte Byte `$X` der IP-Adresse der eintreffenden Meldung nicht der traditionellen Broadcastsuffix 255 entsprechen darf .

Durch diese beiden Erweiterungen ergeben sich vielfältige Möglichkeiten zur kontextsensitiven 1:1-Transformation von Ereignismeldungen, so etwa bei der Anonymisierung/Pseudonymisierung (z. B. Verschleierung von Netztopologien, Herstellern/ Produkten), aber auch bei der Normalisierung.

## 2.2 Beispielproblem Redundanzfilterung

Eine konsequente Erweiterung dieser Technik stellt die Redundanzfilterung dar, die eine *n*:1-Filterung zur Aufaggregation mehrerer, zeitlich eng beieinander liegender, gleichförmiger Meldungen durchführt und durch das Transformationsresultat geeignet repräsentiert. Die wichtigste Erweiterung des XSL-Prozessors ist hier die Unterscheidung zwischen Transformationsregeln für *absolute Matchings* und für *relative Matchings*. Erstere beschreiben die Bedingungen für Ereignismeldungen, bei denen der Aggregationsvorgang beginnt, während letztere die Bedingungen für die darauf folgenden, aufzuaggregierenden Meldungen (i. d. R. als ähnlich zu betrachtende Ereignisse) definieren. Weitere Details hierzu finden sich in [4] und [5].

# 3 Der Kombinationsdetektor

Die Erkennung von vordefinierten Mengen miteinander korrelierter Ereignismeldungen kann ebenfalls als Transformationsprozess formuliert werden: Findet sich in einer Menge von *n* Ereignismeldungen eine zuvor spezifizierte Kombination, so wird eine zusätzliche, entsprechend hoch priorisierte Meldung erzeugt; d.h. es handelt sich um eine *n*:(*n*+1)-Transformation. Die folgenden Abschnitte erläutern zunächst das Verfahren und das Implementierungsprinzip. Danach wird ein konkretes Beispiel diskutiert und im Anschluss werden Optimierungsmöglichkeiten aufgezeigt.

## 3.1 Übersicht

Als Grundlage für die Kombinationserkennung dienen in XSL definierte Transformationsspezifikationen (*Sheets*), die die gesuchten Kombinationen spezifizieren. Entsprechend der Anzahl der Bedingungen, die für die Erfüllung einer Kombination erforderlich sind, beinhalten diese jeweils weitere ineinander geschachtelte Templates, die unter Zuhilfenahme der eintreffenden Ereignismeldungen generiert und unter Berücksichtigung frei definierbarer Übergangsbedingungen aktiviert werden. Wird auf diese Weise eine vollständige Kombination erkannt, so wird im Weiteren keine neue Transformationsvorschrift bzw. Regel, sondern eine Ergebnismeldung oder Warnung generiert. Die Anzahl der aktiven Regeln wird neben entsprechenden Grenzwerten im Wesentlichen durch die Gültigkeitsdauer der einzelnen Regeln gesteuert.

## 3.2 Implementierung

Bereits während der Entwurfsphase wurde besonders darauf geachtet, dass die Leistungsfähigkeit des Kombinationsdetektors trotz Verwendung der ressourcenaufwendigen XSLT-Technologie so hoch wie möglich ist. Einerseits wird dies dadurch erreicht, dass zu einem Zeitpunkt nur die jeweils anstehenden Matchings durchgeführt werden, die zur Vervollständigung einer Kombinationsregel notwendig sind. Dies wird durch die *dynamische Just-in-Time-Regelgenerierung* bewirkt, bei der erst zum Zeitpunkt eines erfolgreichen Matchings die Bedingungen für das jeweils nächste anstehende Matching in den entsprechenden Datenstrukturen abgelegt werden. Wie schon bei der beschriebenen Redundanzfilterung werden nicht alle auf die einzelnen Regelkriterien zutreffenden Meldungen bis zur vollständigen Erfüllung einer Kombinationsregel gespeichert, sondern nur die Daten der Meldungen im Speicher vorgehalten, die zur etwaigen Generierung einer

späteren Warnmeldung erforderlich sind. Die eigentlichen Elementarmeldungen werden unmittelbar weitergeleitet.

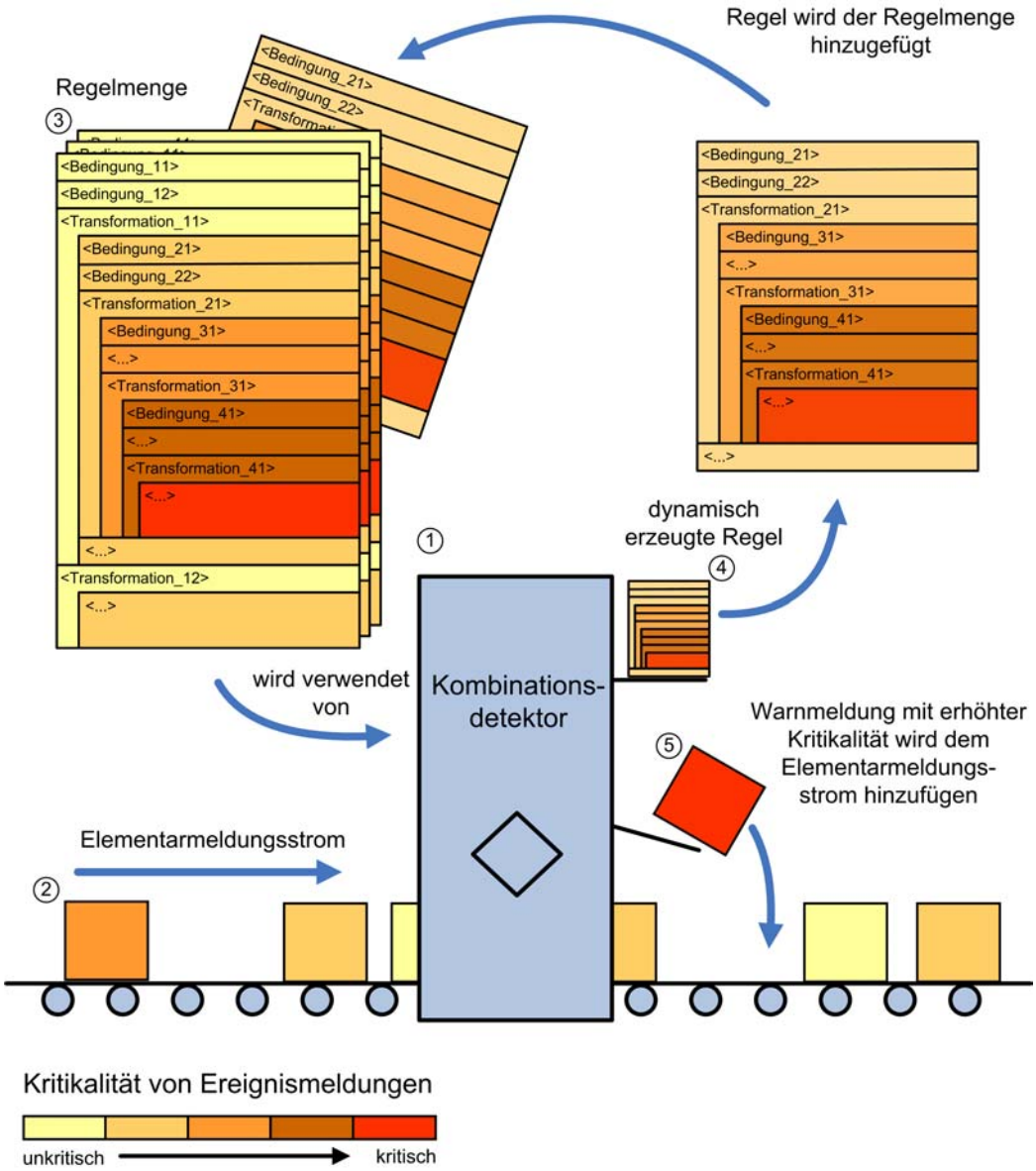


**Abbildung 1:** Kombinationsdetektor - Prinzip der dynamischen Regelgenerierung

Abbildung 1 verdeutlicht den Zusammenhang: Der zum Kombinationsdetektor (1) geführte Elementarmeldungsstrom (2) enthält Meldungen mit unterschiedlicher - meist geringer - Kritikalität, die im Bild durch die helle, gelbe bis orange Einfärbung gekennzeichnet sind. Eintreffende Meldungen werden grundsätzlich immer unverändert durch den Kombinationsdetektor geleitet, so dass keinerlei Informationen verloren gehen. Unter Verwendung der Regelmenge (3) werden die Eingangsmeldungen vom Detektor analysiert und bei erfolgreichem Matching werden auf Basis der aktuell verwendeten Regel und den überprüften Meldungen neue Regeln generiert (4). Anschließend werden die neu spezifizierten Regeln der Regelmenge hinzugefügt, wodurch alle zukünftig eintreffenden Meldungen zusätzlich gegen diese überprüft werden. Die dynamische Generierung von Regeln wird so lange fortgeführt, bis eine Regel zutrifft, die keine weitere neue Regel, sondern eine finale Warnmeldung erzeugt. Dies bedeutet, dass eine vollständige Kombination erkannt wurde, deren Auftreten den nachfolgenden Systemen gemeldet werden muss. Die

Warnmeldung (5) wird daher mit einer erhöhten Kritikalitäts-Einstufung versehen (im Bild dunkelrot gekennzeichnet) und dem Meldungsstrom hinzugefügt.

## 3.3 Einsatzbeispiel

Anhand eines konkreten Einsatzbeispiels soll nachfolgend veranschaulicht werden, wie ein Korrelations-Sheet aufgebaut ist und wie die dynamische Regelgenerierung im Einzelnen vonstatten geht. Kern des Verfahrens ist die Erkennung gleichförmiger Mengen von Meldungen mittels der so genannten *relativen Matchings*, die bei der Implementierung der Redundanzfilterung eingeführt wurden. Im Verarbeitungsprozess sind diese relativen Matchings immer dem absoluten Matching nachgeschaltet und können aufgrund frei definierbarer Attribute die Ereignismeldungen in beliebige Untergruppen aufteilen bzw. kategorisieren.

Es sei in diesem Beispiel folgendes Verhalten des Angreifers zu erkennen:

1. Der Angreifer verwendet ein Tool zur Ermittlung des Administrator-Passwortes auf dem Opferrechner (Wörterbuchangriff – Überprüfen häufig verwendeter Kennungs-/ Passwortkombinationen).
2. Der Angreifer meldet sich mit dem ermittelten Passwort an und importiert sein nötiges „Werkzeug“ auf den Opferrechner.
3. Der Angreifer startet ein Skript zum schnellen Versenden von Spam-Nachrichten (unerwünschte E-Mails) mittels SMTP vom Opferrechner aus.

Die von System zu erkennende Folge von Einzelmeldungs-Signaturen sieht wie folgt aus:

1. Mehr als 5 mal wird in kurzer Abfolge (10 Sekunden) ein nicht erfolgreicher Verbindungsaufbauversuch zum SSH-Einwahldaemon des Opferrechners registriert.
2. Unmittelbar nach Erfassen der 1. Ereignismeldung meldet der SSH-Einwahldaemon einen erfolgreichen Einwahlversuch.
3. Vom Opferrechner (der nicht zum Versand von SMTP-Nachrichten außerhalb des lokalen Netzerwerkes legitimiert ist) gehen nach Ablauf einer gewissen Zeitspanne in schneller Abfolge SMTP-Verbindungen aus.

Die dazugehörige Transformationsspezifikation sieht wie folgt aus:

```
(01) <?xml version="1.0"?>
(02) <IDMEF-Message xmlns:xsl="http://www.w3.org/1999/XSL/Transform"

(03)   <correlation initialMatcher="true".../>

(04)   <Alert>
(05)     <Target>...
(06)       <address>(.*)$TARGET$</address>
(07)     </Target>
(08)    <Source>...
(09)      <address>(.*)$SOURCE$</address>...
(10)    </Source>...
(11)    <Classification>...
(12)      <name>SSH user login failed</name>
(13)    </Classification>
(14)  </Alert>

(15)  <relMatch deltaT="10000" threshold="5">
(16)    <IDMEF-Message>
```

```
(17)      <Alert>
(18)        <Target>...
(19)          <address><xsl:value-of select="$TARGET"/></address>
(20)        </Target>
(21)        <Source>...
(22)          <address><xsl:value-of select="$SOURCE"/></address>
(23)        </Source>
(24)      </Alert>
(25)    </IDMEF-Message>
(26)  </relMatch>

(27)  <transform  finalMatcher="false">
(28)    <IDMEF-Message>

(29)      <correlation initialMatcher="false" deltaTCorr="60000"/>

(30)      <Alert>
(31)        <Target>...
(32)          <address><xsl:value-of select="$TARGET"/></address>
(33)        </Target>
(34)        <Source>...
(35)          <address><xsl:value-of select="$SOURCE"/></address>
(36)        </Source>
(37)        <Classification>...
(38)          <name>SSH user login$</name>
(39)        </Classification>
(40)      </Alert>

(41)      <relMatch threshold="1"...>
(42)        <IDMEF-Message/>
(43)      </relMatch>

(44)      <transform finalMatcher="false">
(45)        <IDMEF-Message>

(46)          <correlation initialMatcher="false" ... deltaTCorr="300000"/>

(47)          <Alert>
(48)            <Source>...
(49)              <address><xsl:value-of select="$TARGET"/></address>
(50)            </Source>
(51)            <Classification>...
(52)              <name>SMTP access from unauthorized node</name>
(53)            </Classification>
(54)          </Alert>

(55)          <relMatch threshold="5" deltaT="5000">
(56)            <IDMEF-Message>
(57)              <Alert>
(58)                <Target>...
(59)                  <address><xsl:value-of select="$TARGET"/></address>
(60)                </Target>
(61)              </Alert>
(62)            </IDMEF-Message>
(63)          </relMatch>

(64)          <transform finalMatcher="true">
(65)            <IDMEF-Message>
(66)              <Alert>
(67)                <Target>...
(68)                  <address><xsl:value-of select="$TARGET"/></address>
(69)                </Target>
(70)                <Source>...
```

```
(71)                  <address><xsl:value-of select="$SOURCE"/></address>
(72)                </Source>
(73)                <Classification>
(74)                  <name>CORRELATION ### Spammer-Combination detected!</name>
(75)                </Classification>
(76)                <Assessment>
(77)                  <Impact severity="high"/>
(78)                </Assessment>
(79)                <AdditionalData meaning="priority"
                      type="integer">6</AdditionalData>
(80)                <AdditionalData meaning="details" type="string">A spammer at
                      <xsl:value-of select="$SOURCE"/> is abusing <xsl:value-of
                      select="$TARGET"/> after a successful dictionary/brute
                      force attack.</AdditionalData>
(81)              </Alert>
(82)            </IDMEF-Message>
(83)          </transform>

(84)          <errorTransform>...</errorTransform>
(85)        </IDMEF-Message>
(86)      </transform>
(87)      <errorTransform>...</errorTransform>
(88)    </IDMEF-Message>
(89)  </transform>
(90)  <errorTransform>...</errorTransform>
(91)</IDMEF-Message>
```

Dieses XSL-Sheet bildet die Korrelation wie gefordert ab und beinhaltet gleichzeitig die zu generierende Ergebnismeldung. Nachfolgend werden die einzelnen Abschnitte des Sheets erläutert:

- Zeilen (01) und (02) sind Kopfzeilen des XSL-Sheets mit der Definition des Namespaces `xsl`.
- Zeile (03) bildet den Anfang der **ersten** Korrelations-Stufe. Der Parameter `initialMatcher="true"` sorgt dafür, dass diese Stufe persistent bleibt, d. h. die Lebensdauer dieser Stufe ist nicht begrenzt.
- Zeilen (4)-(14) beinhalten das initiale (absolute) Matching-Template. Hier werden die beiden freien Variablen `$TARGET$` und `$SOURCE$` gespeichert. Bedingung für das absolute Matching ist hier die Beschreibung `"SSH user login failed"`.
- In das relative Matching-Template (Zeilen (18)-(26)) werden die Werte für `$SOURCE` und `$TARGET` übertragen, wodurch genau eine Quell- und eine Zielangabe als eine Gruppe spezifiziert werden. Das heißt, dass somit für jedes Quelle-Ziel-Paar ein relativer Matcher angelegt wird. Der Parameter `deltaT="10000"`, zusammen mit dem Parameter `threshold="5"`, beschreiben quantitativ die Häufigkeit der Meldungen, die für die Generierung der nächsten Korrelationsstufe erforderlich ist. Wird im Zeitraum `deltaT` der Threshold-Wert nicht erreicht, so wird der relative Matcher für das entsprechende Angreifer-Opfer-Paar verworfen. Somit ist die erste Einzelmeldungs-Signatur vollständig beschrieben.
- Die Zeilen (27)-(89) beschreiben die Transformationsvorschrift der ersten Matching-Stufe, deren Ergebnis die zweite Stufe (Zeilen (44)-(86)) und die darin enthaltene Transformationsvorschrift der dritten Stufe (64)-(83) ist. Die Rekursion endet bei der dritten Stufe, da der Parameter `finalMatcher="true"` in Zeile (64) gesetzt ist. Damit wird festgelegt, das dass entstehende Transformations-Template der letztlich auszugebenden Meldung entspricht.

- Zeile (29) zeigt den Beginn der **zweite** Korrelations-Stufe an. Die Parameter `initial Matcher="false"` sowie `deltaTCorr="60000"` sorgen dafür, dass diese Stufe nach 60 Sekunden gelöscht wird. Auf diese Weise lässt sich das Kriterium des unmittelbaren zeitlichen Folgens des zweiten auf den ersten Punkt der Signaturbeschreibung umsetzen.
- Im Alert (30)-(40) wird eine Meldung mit speziellem Quelle-Ziel-Paar (ermittelt in der ersten Stufe) gesucht, dessen Beschreibung den Wert `"SSH user login"` enthält.
- Der relative Matcher (41)-(43) beinhaltet keine weitere Spezifizierung, da einzig eine Meldung Auslöser für die Generierung der dritten Korrelations-Stufe notwendig ist. Somit ist die zweite Einzelmeldungs-Signatur vollständig beschrieben.
- Zeile (46) beschreibt den Beginn der **dritten** Korrelations-Stufe. Die Parameter `initial Matcher="false"` sowie `deltaTCorr="300000"` sorgen dafür, dass diese Stufe nach 5 Minuten gelöscht wird. Auf diese Weise wird die Bedingung „nach Ablauf einer gewissen Zeitspanne“ im dritten Punkt der Signaturbeschreibung realisiert.
- Im nachfolgenden Alert (47)-(54) wird eine Meldung vom ermittelten Opfer-Rechner gesucht, dessen Beschreibung den Wert `"SMTP access from unauthorized node"` enthält.
- Der relative Matcher (55)-(63) ermittelt durch die beiden Parameter `threshold="5"` und `deltaT="5000"` die Häufigkeit der SMTP Meldungen, wodurch die dritte Einzelmeldungs-Signatur vollständig beschrieben ist.
- Die Zeilen (65)-(83) beschreiben die **zu emittierende Warnmeldung**. In dieser Meldung werden neben den Adressen des Angreifers und des Opfers, der Priorität und dem Schweregrad des Ereignisses, detaillierte Informationen über den erfolgten Angriff angegeben (Zeile (80)).
- Die Zeilen (64), (87) und (90) beschreiben eine Transformationsvorschrift, die im Fehlerfall der entsprechenden Korrelations-Stufe emittiert werden können.

# 4 Optimierung

Über die bereits beschriebenen Maßnahmen zur Vermeidung von Speicherverbrauch hinaus (dynamisch erzeugte Transformationsregeln, sofortiges Weiterleiten von analysierten Einzelmeldungen etc.) wurden weitere Optimierungsansätze verfolgt.

So wird beispielsweise die Analyse einer möglichen Korrelation zeitlich begrenzt. Dies ist sicherlich eine sehr einschränkende Maßnahme, da somit eine Korrelation, die sich über einen sehr großen Zeitraum erstreckt, nicht detektiert werden kann. Die zeitliche Begrenzung betrifft nur die dynamisch erzeugten Regeln, die initialen Regeln sind von dieser Maßnahme ausgeschlossen und somit immer aktiv. Eine zeitliche Begrenzung alleine reicht jedoch nicht aus um zu vermeiden, dass zu viele Regeln gleichzeitig aktiv sind. Um dieses Problem einzuschränken, wurden daher Grenzwerte vorgegeben, bei deren Überschreitung Fehlermeldungen in den Meldungsstrom emittiert werden können. Diese Fehlermeldungen werden im XML-Sheet der entsprechenden Regel angegeben und sind somit vollkommen frei definierbar.

Da die Meldungs-Verarbeitung im hier beschriebenen Verfahren, im Wesentlichen auf der Anwendung eines XSL-Prozessors basiert, ist es natürlich von besonderer Bedeutung, nur wirklich notwendige Transformationen bzw. Matching-Prüfungen durchzuführen. Um auch

dieser Forderung nachzukommen, wurde die Möglichkeit geschaffen, mehrere Korrelationen in einem Korrelations-Sheet zusammenzufassen. Korrelationen kann man weitestgehend als eine Folge von Bedingungen bezeichnen, die als *Korrelations-Zweig* (Correlation Branch) bezeichnet wird. Ein Korrelations-Zweig besteht somit aus aneinander gereihten alternativen Bedingungen oder auch *Korrelations-Stufen*. Die Zusammenfassung von Korrelations-Zweigen gestaltet sich derart, dass Korrelationen, deren Anfangsbedingungen identisch sind, zu einem *Korrelations-Baum* (Correlation Tree) vereinigt werden. Somit ist neben der gesteigerten Leistungsfähigkeit auch die Konfiguration der Regeln wesentlich flexibler.

Die Zusammenfassung der Korrelationen zu den oben genannten Korrelations-Bäumen verhindert jedoch nicht, das u. U. trotzdem identische Regeln in verschiedenen Korrelations-Sheets unnötiger Weise überprüft werden. Um identische Regeln zu erkennen, werden daher Hash-Werte (Checksummen) für alle Regeln - auch für dynamisch erzeugte - berechnet. Fällt beispielsweise die Übereinstimmungsprüfung einer neu eintreffende Meldung gegen eine Regel negativ aus, so wird deren Hash-Wert für die Dauer der Analyse dieser einen Meldung gespeichert und alle Regeln, die ebenfalls den gleichen Hash-Wert besitzen, automatisch nicht auf diese Meldung hin überprüft. Somit muss eine Regelmenge nicht manuell auf doppelte Übereinstimmungsprüfungen hin untersucht und werden, weil der Detektor identische Regeln erkennt und die entsprechenden Überprüfungen unterdrückt.

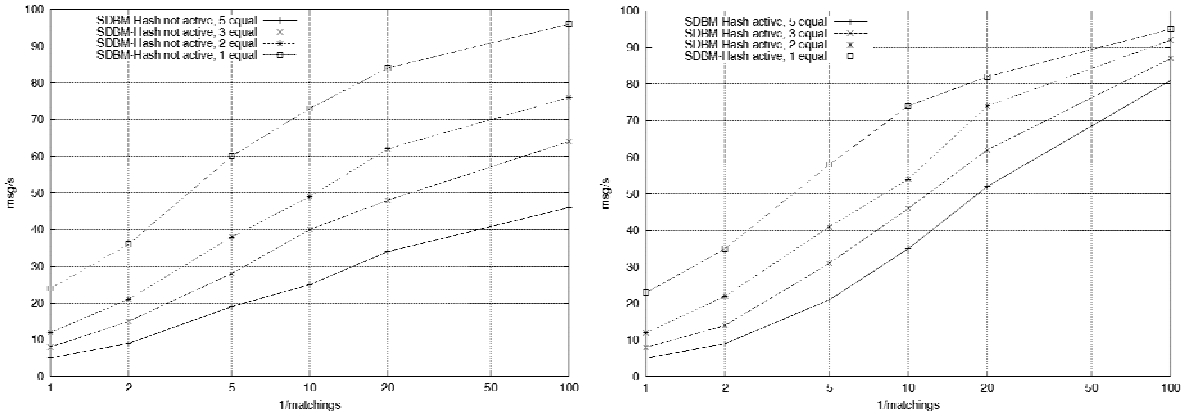


**Abbildung 2:** Anzahl der verarbeiteten Meldungen bei mehreren identischen Transformationsregeln ohne (links) und mit (rechts) verwendetem Hash-Test.

Abbildung 2 skizziert den Gewinn an Verarbeitungsgeschwindigkeit (gemessen in der Anzahl transformierter Meldungen pro Sekunde). Auf der Abszisse ist der Anteil des Eingabe-Meldungsaufkommens angetragen, bei dem eine Übereinstimmungsprüfung gegen die Matching-Templates des verwendeten Regelwerks positiv ausfällt („1/matchings“). Die einzelnen Kurven entsprechen dem Durchsatz der Transformationsstufe bei keiner bzw. 2-5 identischen Regeln im betrachteten Regelwerk. Der Leistungsgewinn bei steigender Anzahl identischer Regeln ist offensichtlich (z. B. 72 statt 46 transformierte Meldungen pro Sekunde bei einem Anteil von 1/100 mit den Matching-Templates übereinstimmender Meldungen und 5 identischen Regeln).

# 5 Vergleich mit anderen Arbeiten

Die Untersuchung von Ereignismeldungs-Korrelationen gehört zu den in der Literatur seit Jahren sehr ausgiebig untersuchten Fragestellungen. So führt beispielsweise das in [6] beschriebene Clustering von Meldungen eine Zuordnung mehrerer Meldungen zu einer einzigen durch, die als Ursache der gemeldeten Ereignisse betrachtet werden kann (*Root Cause*). In [7] werden Ereigniskombinationen zusätzlich mit externen Bedingungen bezüglich der Konfiguration der betrachteten Netze verknüpft.

Die dort und in vielen weiteren Veröffentlichungen beschriebenen Methoden arbeiten vorwiegend auf entsprechenden Datenbanken und fallen daher in die Kategorie „Offline-Verfahren“. Sie sind wegen ihrer Fähigkeit, auch zeitlich weit zurückliegende Ereignismeldungen in den Erkennungsprozess mit einzubeziehen, mächtiger als das hier beschriebene Online-Verfahren.

Mit der Online-Variante ist es dagegen möglich, entsprechende Erkennungskomponenten an beliebigen Orten der Sicherheits-Infrastruktur im Meldungsstrom zu platzieren, da kein Datenbankzugriff benötigt wird. Dadurch ist eine echte Realzeit-Detektion – zumindest von zuvor spezifizierten – Kombinationen möglich, die die übrigen Erkennungsverfahren in idealer Weise ergänzt.

# 6 Zusammenfassung und Ausblick

Das in diesem Beitrag beschriebene Verfahren zur Online-Erkennung von Mengen miteinander korrelierter Ereignismeldungen ermöglicht eine Realzeit-Detektion zuvor spezifizierter Kombinationen, die etwa für das Vorgehen von Angreifern typisch sind (Angriffssequenzen). Die Kombinationen werden durch miteinander verschachtelte Mengen von Transformationsregeln spezifiziert. Auf Basis von erweiterten XSL-Prozessoren wird der in XML-Format vorliegende Meldungsstrom entsprechend des Regelwerks analysiert. Beim Zutreffen einzelner Regeln werden ggf. weitere Stufen des Analyseprozesses angestoßen, an deren Ende die Erzeugung einer entsprechend hoch priorisierten Warnmeldung steht, die die Erkennung der Kombination signalisiert. Einzelne Meldungen, werden nach ihrer Analyse unverändert im Meldungsstrom belassen, sodass keine Verzögerungen entstehen.

Das beschriebene Verfahren wurde bezüglich seiner Leistungsfähigkeit untersucht und bewertet. Verschiedene Optimierungstechniken konnten unter bestimmten Bedingungen ein Anwachsen des Durchsatzes (Anzahl der transformierbaren Ereignismeldungen pro Zeiteinheit) erzielen.

Voraussetzung für die Einsatzfähigkeit dieses sehr flexiblen Verfahrens ist die Existenz einer entsprechenden Datenbank von Kombinationsregeln. Es wäre denkbar, diese – wie etwa die Signaturdatenbank von Snort [8] – auf Basis von Public-Domain-Software zu erweitern und zu pflegen, sodass der Nutzen der Kombinationserkennung einer breiten Öffentlichkeit zugänglich gemacht werden kann.

Derzeit wird im Rahmen der beschriebenen Forschungsarbeiten ein Demonstrationssystem fertig gestellt, dass den Meldungsfluss verschiedener unabhängiger Domänen zusammenführt und analysiert. Auf dieser Basis soll die Praxistauglichkeit und Robustheit des vorgestellten Verfahrens nachgewiesen werden.

# 7 Literaturverzeichnis

[1] Curry, D. und H. Debar: Intrusion Detection Message Exchange Format - Data Model and Extensible Markup Language (XML) Document Type Definition.
IETF Internet Draft draft-ietf-idwg-idmef-xml-10.txt, IETF IDWG, Januar 2003.

[2] W3C. W3C Recommendation 16: XSL Transformations (XSLT) Version 1.0. http://www.w3.org, 1999.

[3] Lies, M., M. Jahnke, J. Tölle, S. Henkel und M. Bussmann: Ein Intrusion-Warning-System für dynamischen Koalitionsumgebungen. In: 11. DFN-CERT/PCA-Workshop „Sicherheit in vernetzten Systemen“, Hamburg, Februar 2004.

[4] Jahnke, M., J. Tölle, S. Henkel und M. Bussmann: Components for Cooperative Intrusion Detection in Dynamic Coalition Environments. In: Proceedings of the RTO/IST Symposium on Adaptive Defence in Unclassified Networks, Toulouse, Frankreich, April 2004.

[5] Jahnke, M., J. Tölle, M. Lies, S. Henkel und M. Bussmann: Komponenten für kooperative Intrusion-Detection in dynamischen Koalitionsumgebungen. In: Proceedings of the GI/ACM/IEEE Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA'04), Dortmund, Juli 2004.

[6] Julisch, K.: Mining Alarm Clusters to Improve Alarm Handling Efficiency. In Proceedings of the NATO/RTO IST Workshop on Inforensics and Incident Response, Den Haag, Niederlande, Oktober 2002.

[7] Cuppens, F. und A. Miege: Alert Correlation in a Cooperative Intrusion Detection Framework. In: Proceedings of the IEEE Symposium on Research in Security and Privacy, Oakland, CA, Mai 2002.

[8] Roesch, M.: Snort – Lightweight Intrusion Detection System for Networks. In: USENIX ´99 Conference Proceedings, 1999.