Ein Grid-basiertes, föderiertes Intrusion Detection System zur Sicherung der D-Grid Infrastruktur (GIDS)

Informationsmodell inklusive Datenaustauschformat für ein Grid-basiertes IDS (MS 12)

Meilenstein zum Abschluss des Arbeitspakets 4

Autoren:

Dr. Wolfgang Hommel	(Leibniz-Rechenzentrum)
Dr. Nils gentschen Felde	(Ludwig-Maximilians-Universität München)
Felix von Eye	(Leibniz-Rechenzentrum)
Jan Kohlrausch	(DFN-CERT GmbH)
Christian Szongott	(Regionales Rechenzentrum für Niedersachsen)

GEFÖRDERT VOM

Inhaltsverzeichnis

Kapitel 1

Einleitung

Dieses Dokument präsentiert als Ergebnis des Arbeitspakets 4 des Projekts „Ein Grid-basiertes, föderiertes Intrusion Detection System zur Sicherung der D-Grid Infrastruktur“ (GIDS) ein Informationsmodell inklusive Datenaustauschformat für ein Intrusion Detection Systemen (IDS) für Grids. GIDS (`http://www.grid-ids.de`) ist ein Teilprojekt im Rahmen des D-Grid (`http://www.d-grid.de`) und wird vom Bundesministerium für Bildung und Forschung (BMBF, `http://www.bmbf.de`) gefördert. Weitere Projektinformationen und Unterlagen können der Projekt-Webseite entnommen werden.

1.1 Projektbeschreibung

1.1.1 Problemstellung

Im Umfeld von Grids ergeben sich im Vergleich zu konventionellen vernetzten Systemen eine Reihe bisher ungelöster Probleme, die es im Falle des D-Grid zu bewältigen gilt. So begegnet man im Grid-Kontext unter anderem einem sehr dynamischen Umfeld. Die hohe Dynamik an verfügbaren Ressourcen, wie auch dynamische Benutzergruppen, dessen Benutzer in virtuellen Organisationen (VOs) zusammengefasst werden, erfordern individuelle, dynamische Nutzersichten, die sich in den Kontext einer VO einbetten lassen und deren individuelle Bedürfnisse erfüllen. Weiter ergibt sich ein Grid-typisch heterogenes Umfeld, welches ebenfalls auf mehreren Ebenen existiert. Nicht nur im Bereich der Ressourcen ist diese Heterogenität zu erkennen. Auch die eingesetzten Grid-Middlewares und vorhandenen Grid-Dienste zeigen dies. Nicht zuletzt die zum Teil bereits von den beteiligten Organisationen eingesetzten Sicherheitskomponenten und -werkzeuge zur Erkennung von Angriffen sind von unterschiedlichster Art.

Hier ist häufig keine Koppelung bestehender Komponenten möglich und der Grid-weite Austausch von Informationen bezüglich sicherheitsrelevanter Ereignisse wird nicht umgesetzt. Dies ist nicht nur auf die Heterogenität in diesem Umfeld zurückzuführen, sondern auch auf Randbedingungen, wie beispielsweise unterschiedliche Sicherheits- und Informationsverbreitungsrichtlinien („security and information sharing policies“) der beteiligten realen Organisationen. Darüber hinaus bieten Firewalls derzeit keinen umfassenden Schutz für Grids. Aufgrund fehlender Mechanismen zur dynamischen Erkennung und Freischaltung von Kommunikationsanforderungen müssen große Portbereiche zum Teil sogar ohne einschränkende Angabe von IP-Adressen permanent freigegeben werden.

Zurzeit existiert kein Gesamtkonzept für ein kooperatives, Grid-weit föderiertes Intrusion Detection System (GIDS) mit entsprechenden Reporting-Komponenten, das sich in ein Umfeld wie dem D-Grid einbettet. Daher soll ein Konzept für ein GIDS entwickelt, im D-Grid implementiert und in die Produktion überführt werden.

1.1.2 Ziel

Ziel dieses Projekts ist die Bereitstellung eines GIDS-Dienstes für das D-Grid. Hierbei gilt es, soweit wie möglich bestehende Ansätze zu integrieren und ein domänen- und organisationsübergreifendes Gesamtsystem zu entwickeln. Insbesondere die Fähigkeit, mit Virtuellen Organisationen (VO) umzugehen und diese auch als Kunden in Betracht zu ziehen, ist dabei von entscheidender Bedeutung. Die Grundidee ist es, Angriffe durch die kooperative Nutzung und Auswertung von lokalen Sicherheitssystemen zu erkennen. Dazu ist der Austausch von Angriffsdaten und somit deren datenschutzkonforme Aufarbeitung, auch zur Wahrung individuell bestehender Sicherheits- und Informationsverbreitungsrichtlinien, notwendig. In einem kooperativen IDS besteht die Möglichkeit, Angriffe schneller zu erkennen, als dies mit unabhängigen und nur die lokale Sicht berücksichtigenden Sicherheitssystemen möglich ist. Somit kann eine Verkürzung der Reaktionszeit der beteiligten Parteien erzielt werden. Weiter können Vorwarnungen, an zum Zeitpunkt der Erkennung eines Angriffs noch nicht betroffenen Parteien, herausgegeben sowie gegebenfalls präventive Gegenmaßnahmen ergriffen werden.

Eine Auswertung der Daten kann sich zu großen Teilen auf bereits vorhandene Ansätze klassischer IDS stützen. Bei der Auswertung der verfügbaren Datengrundlage ist darauf zu achten, dass VO-spezifische Zugriffsrechte und Befugnisse eingehalten werden. Nach erfolgreicher Auswertung aller verfügbaren Informationen durch ein kooperatives und föderiertes GIDS, unter Beachtung individueller Sicherheits- und Datenschutz-Policies, erfolgt eine Berichterstattung über die erkannten Angriffe auf das Grid oder einzelne beteiligte Partner. Auch hier ist es von Bedeutung, dass eine VO-spezifische Sicht auf die bereitgestellten Informationen realisiert wird. Dazu ist eine Anbindung an die im D-Grid bestehenden VO Managementsysteme zu schaffen. Nach der Entwicklung einer geeigneten Architektur für ein kooperatives und föderiertes IDS in Grid-Umgebungen steht die Implementierung und Produktivführung des Systems. Es soll nach Abschluss der Projektlaufzeit ein produktives Intrusion Detection System als Grid-Dienst im D-Grid zu Verfügung stehen, das sowohl von Ressourcenanbietern als auch von Kunden (VOs, Communities etc.) genutzt werden kann.

1.2 Besondere Herausforderungen an das Datenaustauschformat

Das Projekt GIDS baut darauf auf, bereits vorhandene Sicherheitskomponenten auf Seiten der Ressourcenanbieter zu nutzen und deren Alarmmeldungen mit auszuwerten. Dies ist zum einen für die Akzeptanz des Systems wichtig, da ein Ressourcenprovider seine schon vertraute Sicherheitslösung weiter verwenden kann, zum anderen ist eine gewisse Heterogenität von Sicherheitssystemen sinnvoll, da ein potentieller Angreifer seine Angriffe nicht auf Schwachstellen eines einzigen Sicherheitsprogramms ausrichten kann, um das komplette System attackieren zu können.

Zu diesem Zweck ist es jedoch nötig, dass Nachrichten, die das GIDS-Datenaustauschformat verwenden, herstellerübergreifend von allen Sensoren verstanden werden. Die meisten Industrieformate bieten eine solche Interoperabilität nicht, da durch die Fixierung auf ein proprietäres Format eine erhöhte Kundenbindung erreicht werden kann. Freie Standards haben dieses Problem nicht und werden unter anderem in Open Source-Programmen weitestgehend unterstützt. Weiterhin ist die Gefahr von Patentklagen oder plötzlichen Featureänderungen seitens des Herstellers sehr viel unwahrscheinlicher.

1.3 Struktur des Dokuments

Einführend in Kapitel 2 wird die Wahl des IDMEF-Datenformates motiviert. Weiterhin wird eine grobe Einführung in das IDMEF-Datenformat gegeben. Kapitel 3 wird zum einen in einem Vorgriff das Datenschutzkonzept und die Anonymisierung von personenbezogenen Daten angesprochen und ihre Einbindung in das IDMEF-Format dargestellt. Weiterhin werden Möglichkeiten aufgezeigt, wie architekturspezifische Erweiterungen im IDMEF-Standard eingebunden werden können.

Einen Überblick über die verschiedenen im IDMEF-Format vorhandenen Nachrichtentypen gibt Kapitel 4. Diese Typen sind in zwei Klassen eingeteilt. Der Aufbau der ersten Klasse, der sogenannten `Alert`-Klasse, wird in Kapitel 5 beschrieben. An dieser Stelle wird ebenso erläutert, wie sich die Vorgaben des Standards an das GIDS-Datenmodell angleichen lassen. Die Nachrichten der zweiten Klasse, der `Heartbeat`-Klasse, werden in Kapitel 6 erörtert.

1.3.1 Versionsverwaltung

In diesem Dokument wird das Informationsmodell inklusive Datenaustauschformat spezifiziert. Da sich zur Laufzeit des Projektes während der Implementierungen der Agenten oder anderer Teile eventuell Notwendigkeiten für das Informationsmodell ergeben können, wird dieses Dokument mit einer Versionsnummer versehen. Anhand dieser kann die Aktualität des Dokuments überprüft werden.

Versionsnummer	**Datum**	**Änderung**
1.0	30.06.2010	Erste Version

Tabelle 1.1: Version

Kapitel 2

Grundlagen der Syntax und Semantik des GIDS-Datenmodells

Wie bereits im Grobkonzept [5] beschrieben, bildet das **Intrusion Detection Message Exchange Format (IDMEF)** in [6] die Grundlage für das GIDS Datenaustauschformat. Die Entscheidung für dieses Format wird im ersten Abschnitt begründet, bevor im nächsten Abschnitt die grobe Struktur des Datenformates vorgestellt wird. Eine wichtige Eigenschaft des Formats ist seine Flexibilität hinsichtlich der Anforderungen des GIDS. Darauf wird im abschliessenden Abschnitt eingegangen.

2.1 Auswahl des Datenformates

Für die Auswahl des Datenformates bilden die Anforderungen aus den vorangegangenen Meilensteinen die Grundlage. Wichtige Anforderungen, die insbesondere die Auswahl für das GIDS-Datenformat betreffen, sollen hier noch einmal hervorgehoben werden:

Funktionale Anforderungen:

- **F01:** Unterstützung verschiedener Granularitätsstufen bei der Berichterstattung
- **F02:** Berichterstattung zu qualitativ differierenden Angriffen
- **F03:** Aussagekräftige Informationsaufbereitung
- **F04:** Zugriffsmöglichkeit auf Sensordaten
- **F05:** Variationsmöglichkeit der Informationsquellen/Datenbasis zur Laufzeit
- **F08:** Aggregatbildung
- **F12:** Aussagekräftige Informationsaufbereitung

Nichtfunktionale Anforderungen:

- **N01:** Integrierbarkeit in bestehende Management-Werkzeuge
- **N02:** Interoperabilität
- **N05:** Portabilität
- **N06:** Wiederverwendbarkeit
- **N08:** Einheitliche Schnittstellen
- **N09:** Erweiterbarkeit und Flexibilität
- **N10:** Hohe Leistungsfähigkeit
- **N11:** Skalierbarkeit
- **N13:** Dynamik der Ressourcen

N16: Interoperabilität der Sensoren

Sicherheitsanforderungen:

Kryptographische Anforderungen:

S01: Vertraulichkeit von Daten und Nachrichten
S02: Authentizität von Daten und Nachrichten
S03: Integrität von Daten und Nachrichten
S05: Kanal- oder nachrichtenbasierte Kommunikationssicherung
S06: Schutz der Grid-IDS Daten

Nutzerverwaltung:

S11: Zugriffsbeschränkung auf Informationen

Organisatorische und Datenschutzanforderungen:

D06: Anonymisierungs- und/oder Pseudonymisierungsmöglichkeiten
D07: Durchsetzung des Datenschutzes
D08: Nachhalten historischer Berichte
D09: Archivierung von Sensordaten

Um diese komplexen Anforderungen zu erfüllen, scheidet ein eigenständig neu entwickeltes, proprietäres Datenformat praktisch aus. Beispielsweise kann nicht die Interoperabilität mit anderen Systemen garantiert werden, ohne einen unverhältnismäßig hohen Aufwand bei der Entwicklung und Realisierung zu betreiben.

Die MITRE Corporation hat unter [3] eine Liste von Standards zur Aufzeichnung und Übertragung von sicherheitsrelevanten Meldungen herausgegeben. Diese berücksichtigt sowohl Protokoll-Einträge (Logs) wie beispielsweise vom Unix `syslog` als auch Daten von IDS und dient zum Vergleich mit dem von MITRE vorgeschlagenem Format *Common Event Expression (CEE)*. Allerdings ist dieses Format noch in der Entwicklungsphase und wird noch nicht aktiv unterstützt. Andere Standards in [3] sind entweder veraltet (*Common Intrusion Detection Framework / CIDF* und *Distributed Audit Service / XDAS*) oder sind proprietär von einem Hersteller entwickelt worden und unterstützen keine Produkte anderer Hersteller (*Common Base Event / CBE*, *Security Device Event Exchange / SDEE* und *Common Event Format / CEF*). Das Format *Incident Object Description Exchange Format / IODEF* wurde entwickelt, um Daten von Vorfällen zu übertragen und hat deshalb einen anderen Schwerpunkt.

Praktisch als Standard für den Austausch und die Speicherung von IDS-Daten hat sich IDMEF herausgebildet, das von der *Intrusion Detection Working Group* (IDWG) der *Internet Engineering Task Force* (IETF) unterstützt wird. Dabei hat das Format die folgenden Vorteile: Es wird von einer großen Anzahl existierender Intrusion Detection Systeme betreits verwendet und kann leicht in proprietäre oder neue Systeme eingebunden werden. Weiterhin sind viele Lösungen für den sicheren Transport, die Aggregation und Korrelation von IDMEF-Daten und die Speicherung vorhanden. Aufgrund der feststehenden Syntax und Semantik des Datenformates ist IDMEF sehr gut geeignet, um die Meldungen automatisiert zu bearbeiten und auszuwerten. Damit unterstützt IDMEF bereits die Mehrzahl der vorher aufgezählten Anforderungen und beinhaltet flexible Erweiterungsmöglichkeiten für die restlichen Anforderungen.

2.2 Aufbau von IDMEF

Das IDMEF, das in Abbildung 2.1 illustriert wird, stellt ein einheitliches Nachrichtenformat für den Informationsaustausch zwischen Intrusion Detection Systemen dar. Ziel der Entwicklung eines solchen Formats ist die Flexibilität gegenüber verschiedenen Anwendungen. Dieser Anforderung trägt IDMEF durch eine Repräsentation der Nachrichten in der *Extensible Markup Language* (XML) Rechnung.

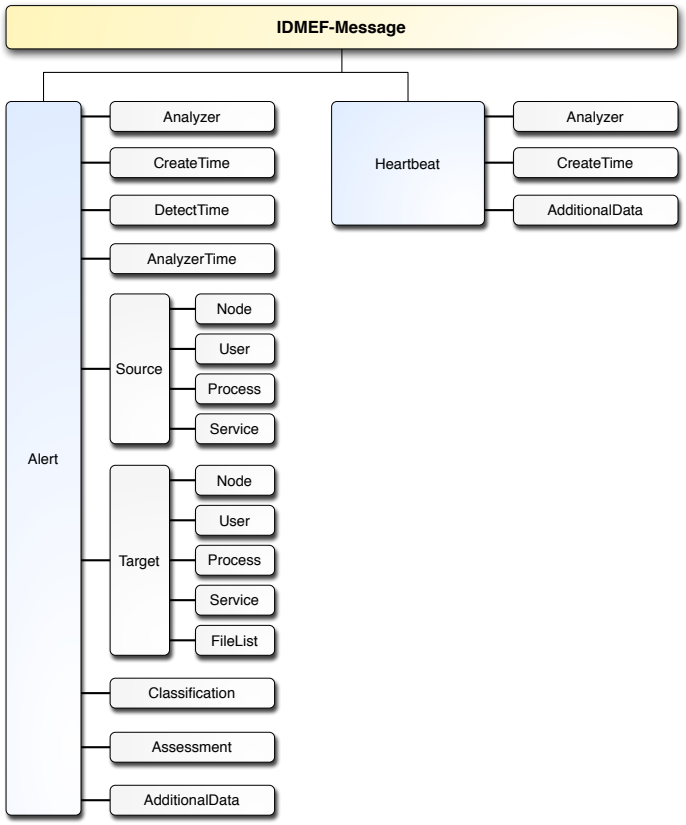

Abbildung 2.1: Das IDMEF-Datenmodell nach [6]

Ein großer Vorteil eines einheitlichen Nachrichtenformats ist die Möglichkeit zur Interoperabilität verschiedener Intrusion Detection Systeme. Anstatt herstellerspezifischer Kommunikationsmechanismen, die in der Regel nur eine Anbindung von Sicherheitskomponenten des gleichen Herstellers erlauben, bietet IDMEF eine Möglichkeit, heterogene Sicherheitskonzepte zu realisieren. Dies wird unter anderem durch flexible Möglichkeiten zur Erweiterung des Formates unterstützt.

Wie aus Abbildung 2.1 hervorgeht, existieren zwei Typen von IDMEF-Nachrichten. Es wird zwischen den sogenannten `Alert-` und den `Heartbeat`-Nachrichten differenziert.

Alert-Nachricht. Dieser Typ von Nachricht wird zur Meldung eines sicherheitsrelevanten Ereignisses versendet. Neben Informationen zum analysierten Element, welches die Meldung generiert hat, sind in einer Alert-Nachricht detaillierte Informationen zu Quelle (`Source`) und Ziel (`Target`) des gemeldeten Ereignisses enthalten. Hierbei ist es insbesondere möglich, Angaben über IP-Adressen und Ports zu machen.

Heartbeat-Nachricht. Heartbeat-Nachrichten sind dazu gedacht, in regelmäßigen Zeitabständen von den einzelnen Komponenten eines gemeinsamen Systems verschickt zu werden. Diese Nachrichten signalisieren die Funktionsfähigkeit der jeweiligen Komponente. Ein Fehlen einer oder mehrerer Heartbeat-Nachrichten weist oftmals auf eine Funktionsstörung oder den Ausfall eines Teils des IDS hin, was wiederum ein Indiz für einen Angriff sein kann.

Die Alert-Nachrichten teilen sich wieder in drei optionale Unterklassen auf, die jeweils den Anlass der Nachricht näher beschreiben:

ToolAlert Diese Unterklasse fügt zusätzliche Informationen über das entsprechende Programm oder die Malware hinzu, von der der Angriff ausging oder die im Zusammenhang

mit dem Angriff steht. Die Informationen umfassen den Namen des Angriffsprogramms oder eine Kommandozeile, durch die der Angriff ausgelöst wurde.

OverflowAlert Eine bedeutende Klasse von Schwachstellen wird durch die fehlerhafte Speicherverwaltung von Programmen in der Programmiersprache C oder einer verwandten Sprache gebildet. Bekanntestes Beispiel dieser Klasse sind Buffer Overflows bei denen ein Buffer mit fester Länge von Angriffsdaten überschrieben wird. Folge davon ist, dass der Fluss des Programms so verändert werden kann, dass eigener Code in dessen Rahmen ausgeführt wird. Mittels OverflowAlert können weitere Informationen über Angriffe auf Buffer Overflows und verwandte Schwachstellen hinzugefügt werden.

CorrelationAlert Angriffe umfassen in der Regel nicht nur einen Schritt sondern mehrere Schritte. Beispiel ist ein Rateangriff auf unsichere Passworte, der in der Regel viele Versuche beinhaltet. Hierbei bietet es sich an, die unterschiedlichen Meldungen bezüglich der missglückten Anmeldeversuche zu einem Alarm zu aggregieren. Weiterhin können mehrere unterschiedliche Meldungen zu einem Angriff durch Korrelation zusammengefasst werden. Diese Möglichkeit wird durch die Klasse CorrelationAlert unterstützt.

2.3 Erweiterungsmöglichkeiten von IDMEF

Das Austauschformat IDMEF bietet über die Klasse `AdditionalData` zwei verschiedene Möglichkeiten für dessen Erweiterung:

- Zuerst können dort Daten in den von IDMEF unterstützten Datenypen abgelegt werden. Diese Variante bietet sich insbesondere bei einfachen Anpassung an.

- Für komplexere Datenstrukturen kann eine eigene XML-DTD festgelegt werden, die die Struktur der Daten angibt. Darüber lassen sich beliebig komplexe Datentypen in die IDMEF-Nachricht einbinden.

2.4 Grundlagen des GIDS-Datenmodells

Wie bereits oben beschrieben, bildet IDMEF die Grundlage für das GIDS-Datenmodell. Daten werden also innerhalb des GIDS auf der Basis dieses Austauschformats transportiert und dessen Struktur bildet die Grundlage für die Speicherung der Daten. Dafür werden die Syntax und Symantik von IDMEF übernommen und die angebotenen Möglichkeiten der Erweiterung zur Anpassung an GIDS ausgenutzt. Die Anpassung ist insbesondere deshalb notwendig, weil IDMEF sehr stark auf der Ebene des einzelnen IDS verwurzelt ist. Dadurch fokussiert das Austauschformat mehrheitlich auf die technischen Aspekte der Sensorik. Damit das Format für das GIDS verwendbar ist, muss dieses an die organisatorischen und architekturbedingten Aspekte angepasst werden:

Organisatorische und rechtliche Aspekte Das Datenschutzkonzept fordert die Einhaltung der rechtlichen Regelungen für den Schutz der Daten. Weiterhin gibt es auf der Seite der Datenzulieferer eine Reihe von Anforderungen, die die Vertraulichkeit und den Schutz von deren Daten betrifft. Dies führt zu einer Anonymisierung oder Pseudonymisierung der Daten. Weiterhin wird die Verwendung der Daten in der Kooperationsvereinbarung festgelegt, die zwischen den Partnern des GIDS abgeschlossen werden. Um diese Aspekte zu berücksichtigen, wird das IDMEF so erweitert, dass pseudonymisierte Daten eingebunden werden können und die Verwendung der Daten spezifiziert werden kann.

Architekturbedingte Aspekte Das GIDS basiert auf einer komplexen Architektur, die in Abb. 3.1 gezeigt ist. Eine der Erweiterungen ist es, dass deren Struktur durch das GIDS-Datenformat wiedergegeben ist. Dies betrifft beispielsweise eine formale Spezifizierung der Sensoren und anderen Komponenten im GIDS.

Das GIDS-Datenmodell wird im Detail in den Kapiteln 4 und 5 beschrieben. Dabei orientiert sich die Beschreibung an der Struktur von IDMEF und geht darauf ein, wie das Format für das GIDS verwendet wird. Wie bereits vorher beschrieben, liegt die spezielle Aufgabe für das Design des GIDS-Datenmodells in der Anpassung von IDMEF. Deshalb liegt der Schwerpunkt auf der Realierung der organisatorischen, rechtlichen und architektonischen Aspekte, die vom föderalen Grid-IDS vorgegeben sind. Vorausgesetzt wird die Syntax und Semantik, wie sie in [6] beschrieben ist.

Kapitel 3

Erweiterungen des IDMEF-Standards für das GIDS-Austauschformat

Der IDMEF-Standard bildet die Grundlage des Austausch- und Datenformates für das föderierte Grid-IDS. Allerdings fokussiert IDMEF im Wesentlichen auf die technische Ebene des Datenaustausches zwischen IDS. Damit ist bereits die überwiegende Mehrzahl der Anforderungen aus dem Kapitel 2 abgedeckt. Es verbleiben aber noch einige organisatorische und rechtliche Aspekte, die sich aus den speziellen Anforderungen des GIDS bilden. Aus diesem Grund werden die Erweiterungsmöglichten von IDMEF ausgenutzt, um diese zusätzlichen Aspekte für das GIDS-Datenformat zu berücksichtigen. Im folgenden Kapitel wird beschrieben, wie die in IDMEF vorgesehenen Erweiterungsmöglichkeiten zur Berücksichtigung dieser Aspekte verwendet werden.

3.1 Anonymisierung und Pseudonymisierung

Die Anonymisierung oder Pseudonymisierung von Daten ist in der Spezifizierung von IDMEF nicht direkt vorgesehen. Allerdings lässt sich das Austauschformat leicht anpassen, so dass dies möglich ist. Dabei können viele Vorteile aus der feststehenden Struktur und detailierten Syntax und Semantik gezogen werden. Zuerst lassen sich alle XML-Entities im IDMEF-Dokument identifizieren, die personenbezogene Daten beinhalten können. Dies betrifft unter anderem die Entities `source` and `target` kann aber auch Rechnernamen und Namen von Benutzern in dem XML `User` Entity betreffen. Für IP-Adressen wurde mit *Crypto-PAn* in [4] ein Rahmenwerk für die Pseudonymisierung entwickelt, das die Netzwerkstruktur der Adressen beibehält. Da die Transformation den Werteraum von IP-Adressen beibehält, lassen sich die transformierten IP-Adressen in den entsprechenden IDMEF-Entities übernehmen. Es sind also keine Änderungen der Syntax von IDMEF notwendig. Mit [2] und [7] wurden weitere Ansätze für die Pseudonymisierung von IDS-Daten vorgeschlagen, die neben IP-Adressen weitere, Stringbasierte Datentypen berücksichtigen. Als weiterer Vorteil können IDMEF-Dokumente leicht automatisch verarbeitet werden, wofür sich eine Vielzahl bereits existierender Programme und Mechanismen verwenden lassen.

Bei der Anpassung von IDMEF müssen aber die folgenden Punkte noch beachtet werden:

- Die orginale Syntax und Semantik von IDMEF sollte nicht ohne zwingende Gründe geändert werden. Als Folge davon sollte bei der Spezifizierung des Datenschutzkonzeptes beachtet werden, dass die pseudonymisierten Daten den entsprechenden Datentyp behalten. Für IP-Adressen bedeutet dies, dass diese auch wieder auf IP-Adressen abgebildet werden. Wie bereits oben beschrieben, würde sich der Ansatz von Crypto-PAn dafür anbieten.

- Es sollte ersichtlich sein, welche Daten pseudonymisiert sind. Dies ist insbesondere für IP-Adressen von Bedeutung, die innerhalb des Werteraums transformiert wurden. Betroffen sind beispielsweise die IDMEF-Entities `source` and `target`. Können diese Metainformationen nicht direkt in den entsprechenden XML-Entities gespeichert werden, bietet sich an, dies unter der Entity `AdditionalData` zu tun.
- Eine Pseudonymisierung lässt sich grundsätzlich wieder rückgängig machen. Dies kann je nach Einsatzzweck gewünscht sein, der vom Datenzulieferer vorgegeben ist. Wie bei der Pseudonymisierung sollte auch der Einsatzzweck der Daten innerhalb des IDMEF-Dokuments spezifiziert werden können. Auch hierfür bietet sich der Entity `AdditionalData` an.

3.2 Architekturspezifische Erweiterungen

Abbildung 3.1 zeigt die grobe Architektur des GIDS. Dafür grundlegend ist die föderale Struktur in der jeder Datenzulieferer eine eigene administrative Domäne verwaltet und die Daten über einen zentralen Bus ausgetauscht werden. Daraus ergeben sich die folgenden Konsequenzen:

- Aufgrund der Autonomie der Datenzulieferer können diese die Verwendung der Daten beeinflussen, die über den Bus exportiert werden. Weiterhin muss berücksichtigt werden, dass ein Datenzulieferer die Löschung der eigenen Daten beantragen kann.
- Für die Analyse der Daten ist es wichtig, deren Quelle unterscheiden zu können. Dies ist insbesondere bei Vorfällen von Bedeutung, die mehrere Systeme unterschiedlicher Einrichtungen betreffen. Um Datenschutzprobleme zu vermeiden, reicht es aus, die Quellen unterscheiden zu können. Deren Identität muss nicht bekannt gegeben werden.
- Um die Datenmenge zu reduzieren und den Aussagegehalt zu erhöhen, werden Daten durch Aggregation oder Korrelation gebündelt. Zwar wird dies grundsätzlich durch IDMEF unterstützt, trotzdem sind vermutlich Erweiterungen notwendig, um besondere Eigenschaften der Aggregation oder Korrelation zu unterstützen.

Mit dem Entity `Analyzer` bietet IDMEF bereits umfangreiche Möglichkeiten, das IDS zu spezifizieren, von dem die Daten stammen. Zusätzliche Daten, die nicht in das Schema passen, können durch das Entity `AdditionalData` hinzugefügt werden. Wie vorher beschrieben, lässt sich dadurch ein eigenständiges XML-Dokument einbinden. Weiterhin kann in diesem Entity der Verwendungszweck der Daten angegeben werden und es können detaillierte Informationen über aggregierte oder korrelierte Daten hinterlegt werden.

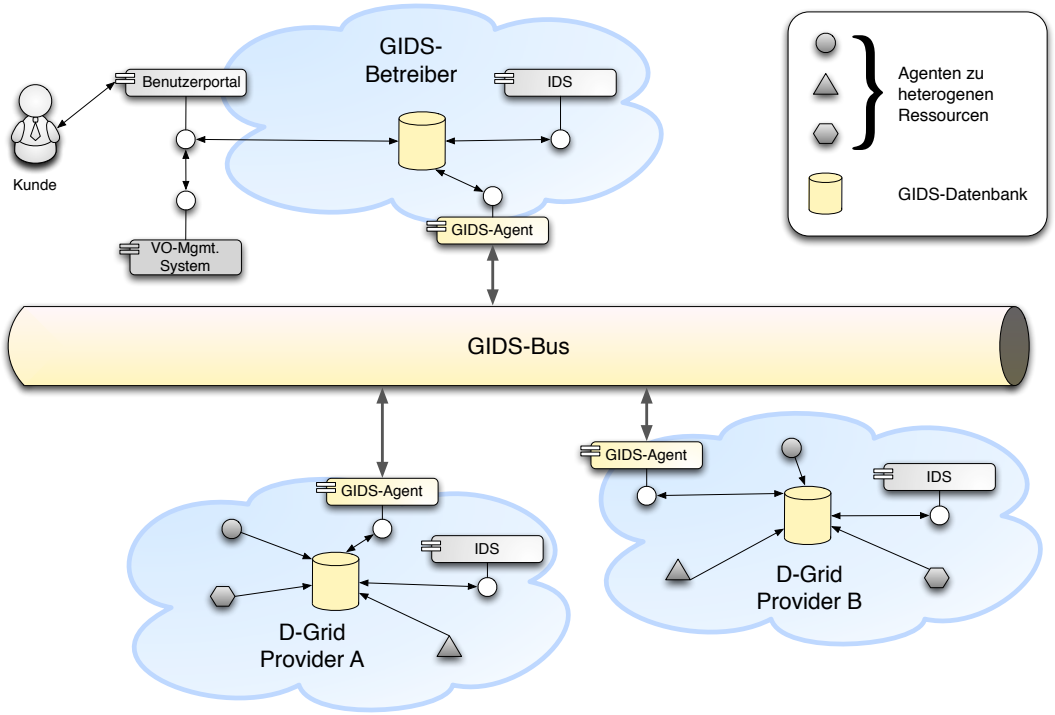

Abbildung 3.1: Grobgranulare Übersicht der Architektur des GIDS

Kapitel 4

GIDS-Nachrichtentypen

Wie bereits vorher beschrieben, teilen sich die IDMEF-Nachrichten in die Klassen `Alert` und `Heartbeat` auf. Weiterhin gibt es von der `Alert`-Klasse drei Unterklassen, deren Motivation die Klassifizierung der Daten auf der Basis der unterschiedliche Quellen ist. In diesem Abschnitt wird darauf eingegangen, wie sich die IDMEF-Klassen in das GIDS-Datenmodell gliedern.

4.1 Nachrichtenklasse Alert

Im Gegensatz zu den `Heartbeats` ist diese Nachrichtenklasse für den Transport von IDS-Meldungen vorgesehen, die sich auf verschiedenen Arten von Angriffen beziehen. Sie teilt sich in drei Unterklassen auf, die aber jeweils einen gemeinsames Grundgerüst haben. Wie bereits vorher beschrieben, wird die Syntax und Semantik im RFC 4765 in [6] für das GIDS-Datenformat übernommen. Dies gilt auch für die Unterklassen des `Alert` Types.

4.1.1 Subklasse ToolAlert

In vielen Fällen werden Angriffe entweder mit Unterstützung von Angriffsprogrammen oder wie im Fall der Internet-Würmer vollständig automatisiert durchgeführt. Um die entsprechenden Details innerhalb einer IDMEF-Nachricht übertragen zu können, wurde die Subklasse `ToolAlert` eingeführt. Damit lassen sich sowohl Angriffsprogramme als auch die Kommandozeile angeben, von der der Angriff ausging. Für die Subklasse `ToolAlert` wird die Syntax und Semantik direkt übernommen.

4.1.2 Subklasse OverflowAlert

Mittels `OverflowAlert` lassen sich Informationen über Angriffe übertragen, die im Zusammenhang mit dem Ausnutzen eines Buffer Overflows oder eine verwandten Schwachstelle stehen. Zwar ist es für das GIDS nicht kritisch, die Informationen bereit zu stellen, trotzdem wird auch hier die Syntax und Semantik direkt übernommen.

4.1.3 Subklasse CorrelationAlert

Eine weitere Unterklasse sind die `CorrelationAlerts`. Dahinter steht die Idee, dass die Nachricht durch Aggregation oder Korrelation von mehreren anderen IDS Alarmen erzeugt wurde. Quelle der Nachricht ist also die Aggregation oder Korrelation von Daten. Da diese Quelle auch bei GIDS eine wichtige Rolle spielt, wird diese auch mit der vorhandenen Syntax und Semantik direkt übernommen. Allerdings wird in der Definition von IDMEF davon ausgegangen, dass ausschliesslich IDMEF-Meldungen aggregiert wurden. Werden für das GIDS-Daten aggregiert, die nicht im IDMEF-Format vorliegen, lassen sich diese Informationen in der `AdditionalData`-Klasse ablegen.

4.2 Nachrichtenklasse Heartbeat

`Heartbeat`-Nachrichten werden innerhalb von IDMEF versendet, um den Ausfall von Systemen zu erkennen. Da dies auch eine wichtige Aufgabe im GIDS ist, wird die Klasse auch für das GIDS-Datenformat übernommen.

Kapitel 5

GIDS-Nachrichten der Angriffsklasse

Um mit Hilfe von IDMEF Alarmmeldungen zu versenden, stellt IDMEF eine `Alert`-Klasse bereit. Kommt es bei einem `Analyzer` (dies können verschiedenste Sensoren sein) zu einem sicherheitsrelevanten Ereignis, welches gemeldet werden soll, so wird eine Nachricht an eine zuvor definierte zentrale Instanz gesendet. Neben Nachrichten, die einzelne Ereignisse anzeigen, gibt es ebenfalls die Möglichkeit, Ereignisse zu gruppieren, um so die Zusammengehörigkeit von Einzelereignissen zu kennzeichnen.

In den folgenden Abschnitten werden zum einen die Bestandteile und die entsprechende Klassenaufteilung nach RFC 4765 [6] beschrieben. In den einzelnen Abschnitten wird zum anderen auf eine mögliche Umsetzung für GIDS eingegangen und dargestellt, welche IDMEF-Klassen jeweils für welche Art von Informationen des GIDS verantwortlich sind.

Die Oberklasse `Alert` setzt sich aus verschiedenen Subklassen zusammen. Hierzu zählen neben Klassen, die für die Grundfunktionalität von IDMEF benötigt werden, auch Klassen für die Aufnahme und Verarbeitung von Elementen, die Zeitpunkte beinhalten, Klassen zur Übermittlung von eingeleiteten Gegenmaßnahmen und weitere Hilfsklassen. Alarmnachrichten können drei verschiedene Gestalten annehmen. Entweder handelt es sich hierbei um einen sog. `ToolAlert`, einen `CorrelationAlert` oder einen `OverflowAlert`. Diese verschiedenen Arten werden in Abschnitt 5.2 erläutert.

5.1 Subklassen von Alert

Die Klasse `Alert` setzt sich aus zahlreichen Subklassen zusammen. `Alert` besitzt nur das Attribut `messageid`, welches eine Nachricht eindeutig auf einem Analyzer identifiziert. Zusammen mit dem später beschriebenen Attribut `analyzerid` kann jede IDMEF-Nachricht im GIDS eindeutig identifiziert werden. Die von `Alert` genutzten Subklassen werden im folgenden erläutert.

5.1.1 Analyzer

Die `Analyzer`-Klasse ist Bestandteil der `Core`-Klassen von IDMEF und ein notwendiger Teil einer IDMEF-Nachricht. Der `Analyzer` repräsentiert die Entität, die einen Angriff erkannt hat und ihn durch die entsprechende Nachricht meldet oder die einen Heartbeat versendet. Pro versandter Nachricht kann nur ein Analyzer benannt werden.

Die `Analyzer`-Klasse hat die acht Attribute:

analyzerid: (optional) Eindeutige Identifizierung des Analyzers

name: (optional) Name, als lesbare Identifizierung des Analyzers

manufacturer: (optional) Name des Herstellers der Hard- bzw. Software, die den Angriff erkannt hat

model: (optional) Modellnummer /-name der Hard- bzw. Software

version: (optional) Version der Hard- bzw. Software

class: (optional) Klasse der Hard- bzw. Software

ostype: (optional) Name des Betriebssystems

osversion: (optional) Version des Betriebsystems

Desweiteren besteht sie aus den aggregierten Klassen `Node`, `Process` und `Analyzer`. `Node` beinhaltet Informationen über den Host bzw. das Device, auf dem sich der `Analyzer` befindet. `Process` enthält Informationen über den Prozess in welchem der Analyzer ausgeführt wird und `Analyzer` enthält Informationen über Analyzer, über welche die Angriffsmeldung verschickt wurde. Diese Angabe ist gerade bei korrelierten Meldungen nötig, um die Daten der Analyzer speichern zu können, die die ursprünglichen Meldungen erzeugt haben.

Für das GIDS-Datenformat sind bei den aggregierten Klassen keine Anpassungen nötig. Bei den Attributen `analyzerid` und `class` wird eine Belegung angestrebt. Dabei soll `analyzerid` eine URI (Uniform Resource Identifier [1]) sein, mit

- **scheme** entspricht *gids*
- **authority** entspricht dem DNS-Namen des Ressourcenproviders
- **path** identifiziert den Analyzer innerhalb der Domäne des Providers eindeutig.

Beispiele für gültige URIs wären `gids://lrz.de/Sec-O-Mat`, `gids://lrz.de/141.84.42.199` oder `gids://rrzn.uni-hannover.de/sensors/01`.
Das `class`-Attribut sollte ein vorgeschriebener Wert aus folgender Auswahl sein:

- NIDS
- Honeypot
- AV-Scanner
- Host-IDS
- System-Log_Analyzer
- Netflow_Analyzer
- RootKit-Scanner
- Other

5.1.2 CreationTime

Bei `CreationTime` handelt es sich um ein notwendiges Element der `Alert`-Klasse. Es bezeichnet das Datum und den Zeitpunkt, an dem der Alarm durch den Analyzer erstellt wurde. Der `CreationTime`-Zeitstempel ist der Einzige, der zwingend in einer IDMEF-Nachricht vorhanden sein muss.

Für das GIDS-Datenformat ist hier keine Anpassung nötig.

5.1.3 DetectTime

`DetectTime` ist ein optionaler Zeitstempel, der den Zeitpunkt der Entdeckung eines Angriffs durch den Analyzer angibt. Sollte es mehrere auslösende Ereignisse geben, so beinhaltet `DetectTime` den Zeitstempel des ersten ausgelösten Alarms. Dieser Zeitpunkt muss nicht zwingend mit dem `CreationTime`-Zeitstempel übereinstimmen, da Alarmmeldungen durchaus auch erst gesammelt zu einem späteren Zeitpunkt verschickt werden könnten. In einem solchen Fall würde sich eine entsprechende Differenz zwischen den beiden Zeiten ergeben.

Für das GIDS-Datenformat ist hier keine Anpassung nötig.

5.1.4 AnalyzerTime

Auch hierbei handelt es sich um einen optionalen Zeitstempel der IDMEF-Nachricht. Dieser gibt die aktuelle Uhrzeit des Analyzers an. Dieser Zeitstempel sollte erst so kurz wie möglich vor dem Versenden der Nachricht gesetzt werden, da er für die Zeitsynchronisation zwischen den verschiedenen Komponenten des Intrusion Detection Systems eingesetzt wird.

Für das GIDS-Datenformat ist hier keine Anpassung nötig.

5.1.5 Source

Dieses Element enthält Informationen über die Quelle, die eine Alarm ausgelöst hat. Bei einem verteilten Angriff kann die `Source`-Klasse auch mehrere Quellen enthalten. Die folgenden aggregierten Klassen setzen sich zur `Source`-Klasse zusammen: Über die Klasse `Node` können Informationen über beliebig viele Hosts bzw. Devices in einer IDMEF-Nachricht mitgeschickt werden, die den Alarm ausgelöst hat (z.B. der Netzwerkname oder die Netzwerkadresse eines potentiellen Angreifers). Die Klasse `User` enthält Daten über den Benutzer, der den Alarm ausgelöst hat. In der `Process`-Klasse werden Informationen gehalten, die den Prozess identifizieren, der den Alarm ausgelöst hat. Schlussendlich sind in `Service` Informationen über Netzwerkservices gespeichert, die an den Ereignissen beteiligt waren.

Neben diesen Informationen sind ebenfalls die drei Attribute `ident`, `spoofed` und `interface` vorhanden.

ident: beschreibt wieder einen optionalen eindeutigen Bezeichner für die Angriffsquelle

spoofed: gibt an, ob die Quelladresse möglicherweise gefälscht worden ist

interface: enthält weitere optionale Angaben zum Interface, auf welchem der Angreifer entdeckt wurde. Dies ist vor allem dann sinnvoll einzusetzen, wenn es sich um einen netzwerkbasierten Analyzer handelt, der mehrere Interfaces besitzt.

Das Attribut `spoofed` hat in der originalen IDMEF-Definition drei möglichen Zustände. In GIDS kann es aber durch Datenschutzbestimmungen oder Informationsverbreitungsrichtlinien nötig sein, IP-Adressen zu anonymisieren oder zu pseudomisieren. Daher wird für das GIDS-Datenmodell ein vierter Wert *9* mit eingeführt, der angibt, dass die IP-Adresse vor der Verteilung pseudonymisiert worden ist.

0 unbekannt

1 IP-Adresse ist wahrscheinlich gefälscht

2 IP-Adresse ist wahrscheinlich nicht gefälscht

9 IP-Adresse wurde pseudonymisiert

5.1.6 Target

Ähnlich zur `Source`-Klasse enthält dieses Element Informationen über mögliche Ziele eines Angriffs, der einen Alarm ausgelöst hat. Entsprechend der `Source`-Klasse sind auch hier mehrere Ziele möglich, wie z.B. bei einem Port-Sweep. `Target` setzt sich aus den fünf Klassen `Node`, `User`, `Process`, `Service` und `File` zusammen. Während die ersten vier genannten analog denen der `Source`-Klasse sind, sind in beliebig vielen `File`-Elementen zusätzlich Informationen über am Angriff beteiligte Dateien enthalten.

Das Attribut `decoy` der `Target`-Klasse hat, wie auch das `spoofed`-Attribut `Source`-Klasse, in der originalen IDMEF-Definition drei möglichen Zustände. In GIDS kann es aber durch Datenschutzbestimmungen oder Informationsverbreitungsrichtlinien nötig sein, Interna der Ressourcenprovider zu anonymisieren oder zu pseudomisieren. Daher wird für das GIDS-Datenmodell ein vierter Wert *9* mit eingeführt, der angibt, dass die Ziel-Angabe vor dem Verteilen unkenntlich gemacht worden ist.

0 unbekannt

1 Ziel ist wahrscheinlich gefälscht

2 Ziel ist wahrscheinlich nicht gefälscht

9 Ziel wurde pseudonymisiert

5.1.7 Classification

Die `Classification`-Klasse ist ein notwendiges Element der IDMEF-Nachricht. Es enthält unter anderem den Namen des Alarms. Im Attribut `ident` befindet sich hierbei ein optionaler, eindeutiger Bezeichner dieser `Classification`. Das notwendige Attribut `name` hingegen enthält einen String, der den Alarm beschreibt. Ebenso sind in `Classification` beliebig viele aggegierte Klassen `Reference` enthalten. `Reference` selbst beinhaltet Verweise auf externe Informationsquellen zum entsprechenden Alarm.

Um eine automatisierte Auswertung der GIDS-Nachrichten zu unterstützen, ist für das Attribut `name` eines der folgenden Werte zu wählen, deren Einteilung an [8] angelehnt ist:

- Viren_und_Wuermer
- Kompromittierung_Grid-Server_und_Middleware
- Privilegieneskalation
- Benutzer-Identitaet
- Grid-IDS
- Denial_of_Service
- Root-Kit
- Trojanisierte_Software
- Bot-Netze
- Missbrauch_von_Ressourcen
- Ausspaehen_von_Informationen
- Manipulation_von_Ressourcen
- Verletzung_der_Grid-PKI
- Zero-day_Exploits

- rawData
- Unknown

Um bei neuartigen Angriffen eine Klassifizierung vornehmen zu können, ist der letzte Wert *Unknown* zu verwenden. Jedoch sollte in diesem Fall die Möglichkeit einer Erweiterung der oben genannten Liste in Erwägung gezogen werden.

5.1.8 Assessment

In der optionalen `Assessment`-Klasse befinden sich neben der Beurteilung des Angriff für das Angriffsziel durch den Analyzer ebenso die Auswirkungen des Angriffs und ergriffene Gegenmaßnahmen. Ebenso ist ein Maß der Konfidenz vorhanden, in der der Analyzer ein selbst geschätzten Wert für die Wahrscheinlichkeit einer korrekten Evaluierung mitliefert. Die Informationen befinden sich in den drei aggregierten Klassen `Impact`, `Action` und `confidence`.

Die Klasse `impact` besitzt die drei Attribute

- `severity` gibt die Schwere des Angriffs an
- `completion` gibt an, ob der Analyzer den Angriff als erfolgreich betrachtet oder nicht
- `type` gibt die Art des Angriffs an (in 6 groben Kategorien)

Die Klasse `action` besitzt das Attribut

- `category` gibt die ergiffene Gegenmaßnahme an (hierbei kann es sich auch nur um eine Benachrichtigung handeln)

Die Klasse `confidence` besitzt das Attribut

- `rating` gibt eine Selbsteinschätzung der Vertrauenswürdigkeit in drei Abstufungen an

Für das GIDS-Datenmodell ist hier keine Anpassung nötig.

5.1.9 Additional Data

Die `AdditionalData`-Klasse enthält alle für das Intrusion Detection System relevanten Informationen, die während des Angriffs gesammelt werden konnten, aber keiner der anderen Klassen des Informationsmodells zugeordnet werden können. Hierbei kann es sich einerseits um besonders detaillierte Daten eines Angriffs handeln. Andererseits kann es für eine erfolgreiche Erkennung/Bekämpfung eines Angriffs erforderlich sein, komplexe Daten zu übertragen. Auch dies ist mit Hilfe der `AdditionalData`-Klasse möglich. Das Element kann die Daten des Datentyps boolean, byte, character, date-time, integer, ntpstamp, portlist, real, string, byte-string und xmltext aufnehmen und ist somit flexibel genug, alle benötigten zusätzlichen Informationen in das Informationsmodell aufzunehmen. Neben den zusätzlichen Daten selbst wird in einem Attribut-String `meaning` beschrieben, welche Bedeutung die enthaltenen Daten haben. Weiterhin bietet die `AdditionalData`-Klasse wie in [6] beschrieben die Möglichkeit, den IDMEF-Standard für beliebige Daten zu erweitern.

5.2 Spezialisierung der Alarmklassen

In Abbildung 5.1 ist die Spezialisierung nach [6] dargestellt. Ein Alarm kann entsprechend entweder ein ToolAlert, ein OverflowAlert oder ein CorrelationAlert sein. Die Bedeutung und Funktion der speziellen Klassen wurde in Kapitel 4 erläutert. In den folgenden Abschnitten soll die technische Umsetzung dargestellt werden.

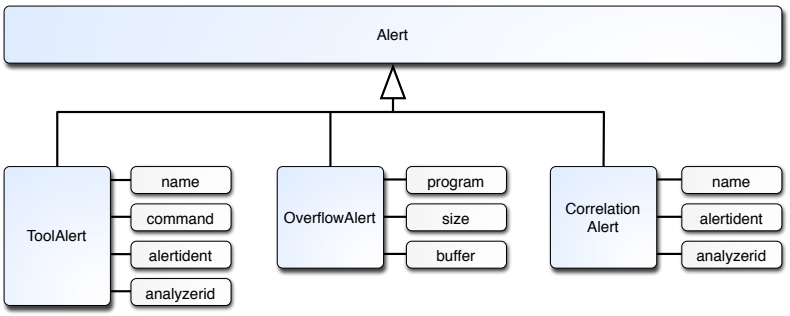

Abbildung 5.1: Spezialisierung der `Alert`-Klasse in IDMEF [6]

ToolAlert

Die Klasse `ToolAlert` besitzt ihrerseits wieder die notwendige Subklasse `name` und die beiden optionalen Klassen `command` und `alertident`. Der Name gibt den Grund für die Gruppierung von Alarmen an, z.B. der Name eines bestimmten Tools. Im `command` wird das Kommando bzw. die Operation gelistet, die vom entsprechenden Tool ausgeführt werden sollte. Das optionale `alertident` beinhaltet eine Liste mit `alertidentifiers`, die zu diesem Alarm gehören. Diese `alertidentifiers` sind nur innerhalb eines Analyzers einmalige IDs. Durch das zusätzliche setzen des `analyzerId`-Attributs von `alertident` besteht die Möglichkeit den Analyzer zu identifizieren.

CorrelationAlert

Ein `CorrelationAlert` dient der Gruppierung von Einzelalarmen. Hierfür wird neben einem Namen, der den Grund für eine Gruppierung angibt (z.B. eine bestimmte Korrelationsmethode), eine Liste von `alertidentifiers`, die in Bezug zu diesem Alarm stehen. Ebenso wie in Abschnitt 5.2 (ToolAlert), wird die `analyzerid` benötigt, um die in der Nachricht enthaltenen `alertidentifiers` eindeutig zuordnen zu können.

OverflowAlert

Ein OverflowAlert beinhaltet neben den obligatorischen Informationen aus der Alert-Klasse zusätzliche Informationen bezüglich Buffer Overflow-Attacken. Dies soll es einem Analyzer ermöglichen, detaillierte Informationen über die Attacke selbst zu verteilen. Ein OverflowAlert besteht aus den drei Klassen `program`, `size` und `buffer`. Im `program`-Teil steht der Name des Programme, welches während der Overflow-Attacke zu starten versucht wurde. Die Größe des Überlaufs wird in `size` festgehalten und `buffer` bietet die Möglichkeit Teile oder auch die Gesamtheit der Daten, die für den Überlauf gesorgt haben, mit zu senden.

5.3 Überblick

Abbildung 5.2 zeigt den detailllierteren Aufbau einer IDMEF-Nachricht. Neben der Erweiterung durch die direkten Subklassen sind ebenfalls die Attribute der einzelnen Klassen eingezeichnet.

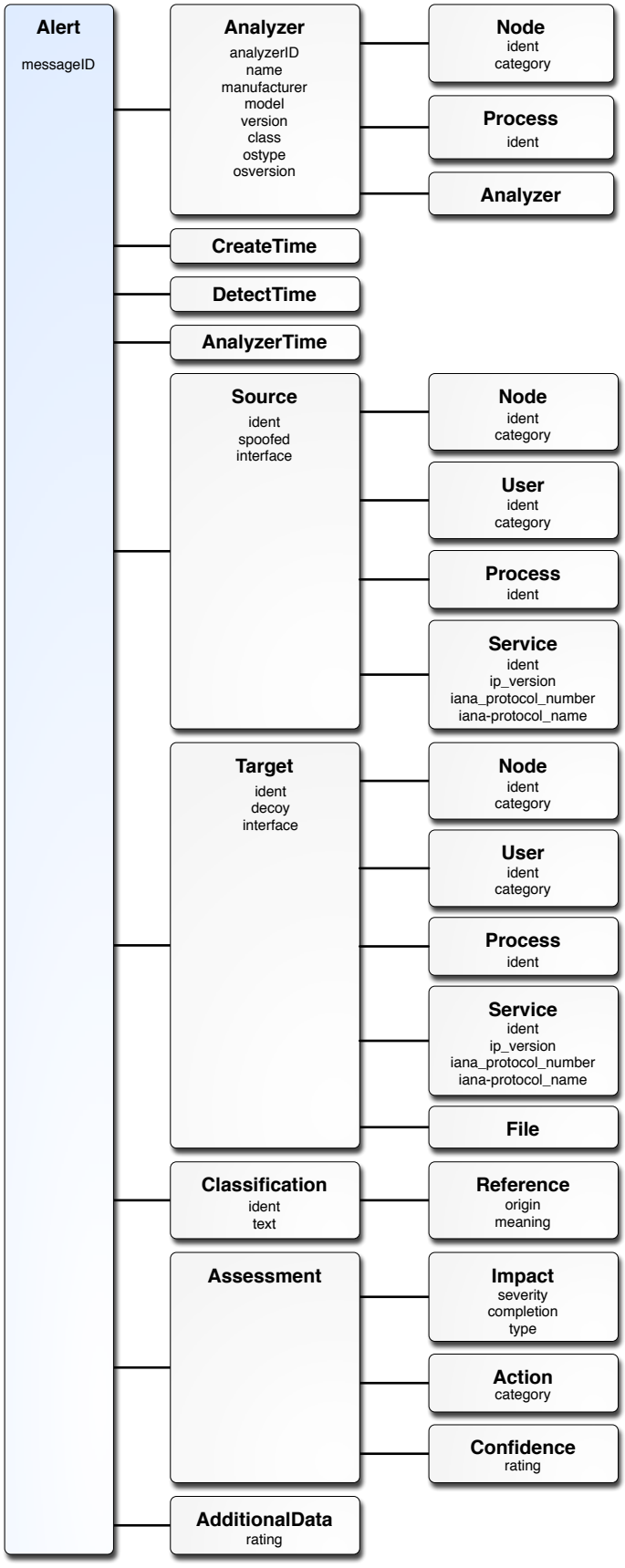

Abbildung 5.2: Das detaillierte IDMEF-Datenmodell der `Alert`-Klasse nach [6]

Kapitel 6

GIDS-Nachrichten der Klasse Heartbeat

In einem Intrusion Detection System muss sichergestellt werden, dass die angeschlossenen `Analyzer` zuverlässig arbeiten. Durch technische Störungen oder Angriffe auf die Analyzer selbst, könnte ein Angreifer versuchen, seine Aktionen zu verschleiern. Um der zentralen Manager-Instanz des GIDS mitzuteilen, dass ein Analyzer weiterhin funktionstüchtig ist, werden sog. `Heartbeat`-Nachrichten verwendet. Das fehlen einer oder vieler aufeinanderfolgender Heartbeat-Nachrichten deutet daraufhin, dass ein Analyzer nicht korrekt arbeitet oder die Netzwerkverbindung gestört ist.

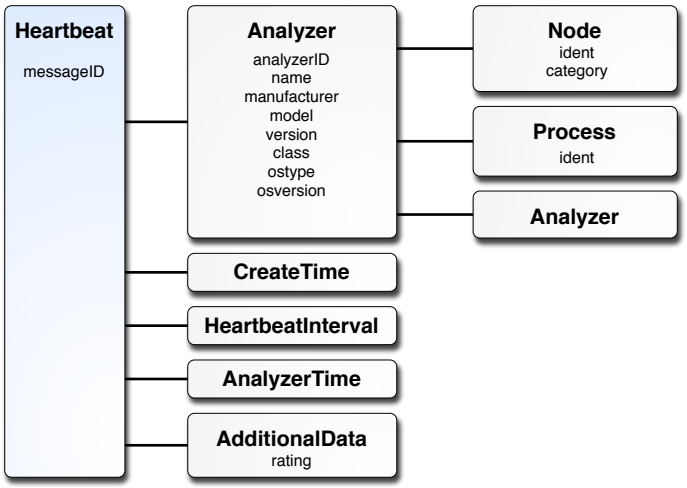

Abbildung 6.1: Das detaillierte IDMEF-Datenmodell der `Heartbeat`-Klasse nach [6]

Der Aufbau einer Heartbeat-Nachricht ist in Abbildung 6.1 dargestellt. Sie besteht im Wesentlichen aus Klassen, die bereits aus der `Alert`-Klasse bekannt sind. Hinzugekommen ist das `HeartbeatInterval`. Die einzelnen Klassen und ihre Funktion innerhalb einer `Heartbeat`-Nachricht werden in den folgenden Abschnitten erläutert.

6.1 Analyzer

Ebenso wie bei einer `Alert`-Nachricht ist die Klasse `Analyzer` zwingend in einer `Heartbeat`-Nachricht erforderlich und gibt an, von welchem Analyzer der Heartbeat stammt. Die enthaltenen Attribute, weitere Subklassen, sowie deren Attribute entsprechen denen der `Alert`-Klasse.

6.2 CreateTime

In `CreateTime` wird festgehalten, zu welchem Zeitpunkt die Heartbeat-Nachricht erzeugt wurde. Wie auch bei der `Alert`-Nachricht, ist dieses Element zwingend erforderlich.

6.3 HeartbeatInterval

Im `HeartbeatInterval`-Element wird dem Manager angezeigt, wie häufig Heartbeats von diesem Analyzer erzeugt werden. Das `HeartbeatInterval` wird in IDMEF in Sekunden angegeben. Anhand dieser Information, kann ein Manager bestimmen zu welchem nächsten Zeitpunkt ein erneuter Heartbeat zu erwarten ist und wie viele Heartbeats im Fehlerfall nicht korrekt bei ihm eingegangen sind.

6.4 AnalyzerTime

Gibt die aktuelle Uhrzeit des Analyzers an. Für weitere Informationen, siehe Abschnitt 5.1.4.

6.5 AdditionalData

Auch hier bietet IDMEF in diesem Element die Möglichkeit, zusätzliche Informationen mitzuliefern, die inhaltlich in keine der anderen Element passt. Für weitere Informationen, siehe Abschnitt 5.1.9.

Kapitel 7

Zusammenfassung und Ausblick

In diesem Kapitel wird das Datenformat des föderierten Grid-IDS beschrieben, das für den Austausch der IDS-Nachrichten verwendet wird. Weiterhin bildet das Datenformat die Grundlage für die Speicherung dieser Daten.

Die Auswahl des Formats erfolgte auf der Grundlage der vorher herausgearbeiteten Anforderungen an das GIDS. Es zeigte sich, dass eine vollständig eigenständige Spezifizierung des Datenformates einen extremen Aufwand bedeutet hätte. Für den Austausch von IDS-Daten hat sich inzwischen IDMEF in [6] als Standard etabliert. Viele IDS unterstützen dieses Format und es sind viele Programme und Bibliotheken verfügbar, die bei der Unterstützung oder Anpassung anderer Sensoren Hilfe bieten.

Als entscheidender Vorteil von IDMEF unterstützt dieses Format die Mehrzahl der Anforderungen des GIDS. Aus diesem Grund bildet die Syntax und Symantik des IDMEF [6] das Grundgerüst für das GIDS-Datenformat. Um auch die organisatorischen und spezifischen Aspekte des GIDS zu unterstützen, wird das Format entsprechend mit den von IDMEF bereitgestellten Methoden erweitert. Kern dieses Dokumentes bildet die Beschreibung, wie das GIDS-Datenformat auf der Basis von IDMEF umgesetzt wird und welche Erweiterungen vorgenommen werden.

Da die GIDS-Architektur einem dynamischen Wachstum unterzogen ist, bei dem immer wieder neue Sensoren berücksichtigt werden müssen, wird auch das Datenformat stetig angepasst. Um diese Änderungen nachvollziehen zu können, wird die Historie der Erweiterungen und Ergänzungen am Anfang des Dokumentes beschrieben.

Abbildungsverzeichnis

Literaturverzeichnis

[1] T. Berners-Lee, R. Fielding, U.C. Irvine, and L. Masinter. Uniform Resource Identifiers (URI): Generic Syntax. RFC 2396, August 1998.

[2] Joachim Biskup and Ulrich Flegel. On pseudonymization of audit data for intrusion detection. In *International workshop on Designing privacy enhancing technologies*, pages 161–180, New York, NY, USA, 2001. Springer-Verlag New York, Inc.

[3] MITRE Corporation. Comparison to other efforts: Standards zur aufzeichnung von logs und events. *http://cee.mitre.org/comparison.html*.

[4] Jinliang Fan, Jun Xu, Mostafa H. Ammar, and Sue B. Moon. Prefix-preserving ip address anonymization: measurement-based security evaluation and a new cryptography-based scheme. *Computer Networks*, 46(2):253–272, 2004.

[5] Wolfgang Hommel, Nils gentschen Felde, Felix von Eye, Jan Kohlrausch, and Christian Szongott. Grobskizze einer Architektur. Meilensteinbericht, D-Grid, April 2010.

[6] IETF. The intrusion detection message exchange format (idmef). *http://tools.ietf.org/html/rfc4765*, 2007.

[7] Florian Kerschbaum. Distance-preserving pseudonymization for timestamps and spatial data. In *WPES '07: Proceedings of the 2007 ACM workshop on Privacy in electronic society*, pages 68–71, New York, NY, USA, 2007. ACM.

[8] Helmut Reiser, Nils gentschen Felde, Felix von Eye, Jan Kohlrausch, and Christian Szongott. Anforderungs- und Kritierienkatalog (MS 6). Meilensteinbericht, D-Grid, January 2010.